PARAMETRIZACIÓN DE SERVICIOS EN ALTA DISPONIBILIDAD PARA ECOSISTEMAS DE BIG DATA, UTILIZANDO HERRAMIENTAS DE AUTOGESTIÓN Y AUTOCONFIGURACIÓN

RICARDO FETECUA PUENTES

UNIVERSIDAD CATÓLICA DE COLOMBIA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

BOGOTÁ D.C

2018

PARAMETRIZACIÓN DE SERVICIOS EN ALTA DISPONIBILIDAD PARA ECOSISTEMAS DE BIG DATA, UTILIZANDO HERRAMIENTAS DE AUTOGESTIÓN Y AUTOCONFIGURACIÓN

RICARDO FETECUA PUENTES

Trabajo de Grado para Optar al Título de

Ingeniero de Sistemas

Director

Diego Alberto Rincón Yáñez MCSC

UNIVERSIDAD CATÓLICA DE COLOMBIA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

BOGOTÁ D.C

2018

NOTA DE ACEPTACIÓN

Aprobado por el comité de grado en cumplimiento de los requisitos exigidos por la Facultad de Ingeniería y la Universidad Católica de Colombia para optar al título de Ingenieros de Sistemas.

_____

Jurado

_____

Director

_____

Revisor Metodológico.

Bogotá D.C., Noviembre 2018

AGRADECIMIENTOS

En primer instancia quiero agradecer a Dios por permitirme adquirir conocimientos nuevos en la Universidad Católica de Colombia que me servirán para mi futuro profesional en segunda instancia, a mis padres que durante largas jornadas laborales reunían el dinero para costearme mis estudios, su apoyo incondicional y desinteresados se los agradeceré toda la vida, a todas aquellas personas que durante mi paso por la Universidad me enseñaron muchas cosas y me colaboraron para que esta nueva menta fuera posible, también agradecer a mi tutor el ingeniero Diego Rincón por su gran ayuda en el desarrollo de mi trabajo de grado por transmitir de forma desinteresa su conocimiento, una vez gracias y mil veces gracias a todos ustedes.

TABLA DE CONTENIDO

Pág.

LISTA DE IMÁGENES

GLOSARIO

ALTA DISPONIBILIDAD: Disponibilidad se refiere a la práctica de los usuarios para acceder y usar el sistema. Si un usuario no puede acceder al sistema se dice que está no disponible. El término tiempo de inactividad o tiempo fuera de línea es usado para definir cuándo el sistema no está disponible. Así que Alta Disponibilidad es que el sistema está casi siempre arriba gracias a que de manera anticipada se duplicaron todos los elementos de Servidores, Networking y almacenamiento eliminando los puntos únicos de fallos y a nivel de Software se usan sistemas inteligentes y automatizados que restablecen el servicio[1].

ATRIBUTO: Un atributo es una especificación que define una propiedad de un objeto, elemento o archivo. También puede representar o establecer el valor específico para una instancia determinada de los mismos[2].

BIG DATA: Es un término que refiere el gran volumen de datos, tanto estructurados como no estructurados, que inundan los negocios cada día. Big Data se puede analizar para obtener ideas que conduzcan a mejores medidas y movimientos de negocios estratégicos, tales como bases de datos relacionales y estadísticas convencionales o paquetes de visualización[3].

COOKBOOKS: Éstos son la unidad fundamental de la distribución de configuración y políticas de Chef, pues define todo lo necesario para así poder realizar las modificaciones o instalaciones necesarias[4].

DATABAGS: Existen programas denominados sistemas gestores de bases de datos, abreviado SGBD (del inglés Database Management System o DBMS), que

[1] **disponibilidad, Sistemas de alta. 2018.** repository.udistrila.edu.co. [En línea] 2018. [Citado el: 02 de 07 de 2018.]
[2] **Atributos. 12.** Microsoft. [En línea] 2017, 10 de 10 de 12. [Citado el: 08 de 09 de 2018.] https://docs.microsoft.com/es-es/dotnet/framework/wpf/xaml-syntax-in-detail.

[3]**consiste, En qué. 2017.** PoweData. [En línea] 2 de 07 de 2017. [Citado el: 02 de 09 de 2018.] https://www.powerdata.es/big-data.
[4] **COOKBOOK. 2018.** linke. [En línea] 19 de 11 de 2018. [Citado el: 19 de 11 de 2018.] https://blog.linkeit.com/glosario-terminos-chef-server.

permiten recopilar y posteriormente acceder a los datos de forma rápida y estructurada (DATABAGS, 2017).

NODOS: Los nodos son todos y cada uno de los equipos informáticos que están interconectados y configurados por el servidor. Estos nodos son independientes y pueden ser tanto hardware físico como contenido en la nube. Sea como sea el nodo, éste se sirve de una aplicación conocida como chef-client para comunicarse con el servidor. La destino de esta aplicación no es otra que la de extraer datos del servidor y ejecutar los pasos de configuración para que se obtenga el resultado.

OHAI: Es una herramienta que utiliza el servidor para detectar ciertas propiedades de cada nodo para después proporcionarlos al chef-client durante la ejecución de este.

RECETAS: Las recetas es otro de los términos más reiterados en este mundo, pues son pequeños archivos que definen cómo se debe configurar el servicio, de ahí su importancia. Las recetas se encuentran en lo que se conoce como Cookbooks[5].

RUN-LIST: Es una lista en el que se pone los roles o recetas que queramos que se ejecute, se ejecutarán en el orden que pongamos.

RUTINAS: La programación estructurada es un paradigma de programación orientado a mejorar la calidad y tiempo de desarrollo de un programa de computadora recurriendo únicamente a subrutinas y tres estructuras básicas: secuencia, selección (if y switch) e iteración (bucles for y while); asimismo, se considera innecesario y contraproducente el uso de la instrucción de transferencia incondicional (GOTO), que podría conducir a código espagueti, mucho más difícil de seguir y de mantener, y fuente de numerosos errores de programación[6].

SCRIPT: Archivo de órdenes, archivo de procesamiento por lotes o, cada vez más aceptado en círculos profesionales y académicos, es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano. Los guiones son casi siempre interpretados, pero no todo programa interpretado es considerado

[5] **CHEF.IO. 2017.** RECETAS. [En línea] 15 de 12 de 2017. [Citado el: 5 de 6 de 2018.] https://docs.chef.io/recipes.html.

[6] **estructurada, Programación. 2018.** Wikipedia. [En línea] 19 de 10 de 2018. [Citado el: 20 de 10 de 2018.] https://es.wikipedia.org/wiki/Programaci%C3%B3n_estructurada.

un guion. El uso habitual de los guiones es realizar diversas tareas como combinar componentes, interactuar con el sistema operativo o con el usuario. Por este uso es frecuente que los intérpretes de órdenes sean a la vez intérpretes de este tipo de programas[7].

SERVIDOR CHEF: El Chef Server es el punto central donde se almacenan las recetas de configuración, que no son más que una serie de archivos, los nodos y las definiciones de la estación de trabajo. El servidor que en este caso es Chef, actúa como un Hub de configuración y se encarga de almacenar los CookBooks, las políticas de configuración de las recetas y los nodos que componen la red del servidor[8].

TEMPLATES: El directorio de plantillas o templates es el que se utiliza para gestionar la configuración más compleja. Es capaz de proporcionar archivos de configuración cumplidos, los cuales contienen comandos embebidos en Ruby.

[7] **Scrip. 2017.** webdesarrolladores. [En línea] 01 de 11 de 2017. [Citado el: 25 de 07 de 2018.] http://websarrolladores.com/2018/09/10/script/.

[8] **linke. 2018.** Chef-server. [En línea] 15 de 06 de 2018. [Citado el: 10 de 9 de 2018.] https://blog.linkeit.com/glosario-terminos-chef-server.

ABSTRACT

This project develops a series of parameters on high availability services for Big Data environments using a self-management and self-configuration tool, selecting four services from them on a personal level.

An investigation is carried out on the tool of self-management and on the services analyzing its architecture and the work scheme as a cluster, then the factors that act in a high availability service are studied.

It proceeds to raise the tool of self-management and self-configuration with the components that it needs for its correct functioning as it is the Chef-Server, Chef-Workstation and the Chef-Client, for the case of the Chef-Server the recipes or books are loaded of kitchen that contain the routines to run the selected services, with the Chef-Workstation it controls the recipes and the nodes that are in charge of an organization and the Chef-Client are the different nodes that are needed so that the services work in high availability.

Next, the Chef-Server and its administration console are configured so that the Chef-Workstation can execute its functions. In addition, an organization and access codes are set up between the workstation and the server, guaranteeing Chef's privacy and security. The recipes that contain the Scripts are elaborated in order to install the services in high availability.

Finally, an analysis of the results and comparison of times between the deployment of services individually and deploying them with the help of the self-management and self-configuration tool is made attaching configuration manuals.

Key words: Big Data, clúster, nodes, routines, script, services, high availability.

RESUMEN

Este proyecto desarrolla una serie de parámetros sobre servicios en alta disponibilidad para ambientes Big Data usando una herramienta de autogestión y autoconfiguración, seleccionando cuatro servicios de ellos a nivel personal.

Se realiza una investigación sobre la herramienta de autogestión y sobre los servicios analizando su arquitectura y el esquema de trabajo como clúster, luego se estudia los factores que actúan en un servicio de alta disponibilidad.

Se procede a levantar la herramienta de autogestión y autoconfiguración con los componentes que ella necesita para su correcto funcionamiento como lo es el Chef-Server, Chef-Workstation y el Chef-Client, para el caso del Chef-Server se cargan las recetas o libros de cocina que contienen las rutinas para correr los servicios seleccionados, con el Chef-Workstation se controla las recetas y los nodos que están a cargo de una organización y el Chef-Client son los diferentes nodos que se necesitan para que los servicios funcionen en alta disponibilidad.

A continuación se configura el Chef-Server y su consola de administración para que el Chef-Workstation pueda ejecutar sus funciones. A demás se configura una organización y unas claves de acceso entre la estación de trabajo y el servidor, garantizando privacidad y seguridad de Chef. Se elaboran las recetas que contienen los Scripts para poder instalar los servicios en alta disponibilidad y se cargan por medio del Chef-Workstation levantando el servicio.

Finalmente se hace un análisis de los resultados y la comparación de tiempos entre el despliegue de servicios de forma individual y desplegándolos con ayuda de la herramienta de autogestión y autoconfiguración anexando manuales de configuración.

Palabras clave: Big Data, clúster, nodos, rutinas, script, servicios, alta disponibilidad.

INTRODUCCIÓN

En la actualidad existen herramientas de autogestión y autoconfiguración para el uso de servicios aplicados a ecosistemas de Big Data y diferentes tipos de arquitecturas de software que permiten agilizar los procesos en la área de TI (Tecnología de la información), en ocasiones la falta de conocimiento y la falta destreza de directores de TI, al implementar estas herramientas hacen que los procesos de infraestructura de las diferentes entidades sea un poco tediosas y costosas.

El siguiente proyecto tiene como finalidad parametrizar servicios en alta disponibilidad para ecosistemas de Big Data utilizando la herramienta de autogestión y autoconfiguración Chef para su correcto despliegue.

También analizar el comportamiento de cuatros servicios para ecosistemas de Big Data, partiendo de este análisis se obtendrán lineamientos para la configuración y parametrización de los mismo, con la finalidad de ser implementados los cuatro servicios en alta disponibilidad sobre la herramienta de autogestión y auto configuración.

Permitiendo optimizar los tiempos de ejecución de los diferentes servicios, con esto podemos comprobar el beneficio que se obtiene usando estas herramientas en la actualidad, ya que es una falencia que se tiene en el área de TI de muchas compañías por falta de una buena parametrización de un herramienta de autogestión que en algunos casos no son conocidas por los encargado de las áreas de infraestructura, debido a esto se busca una herramienta la cual sea de comercialización libre y amigable, con documentación actualizada para su implementación y configuración cuando los usuarios desean hacer algún tipo de modificación.

Una vez parametrizados los servicios sobre la herramienta de autogestión y autoconfiguración se procederán a realizar pruebas de despliegue sobre los 4 servicios orientados a ecosistemas de Big Data en alta disponibilidad previamente seleccionados para la posterior demostración de mejoras en tiempos de ejecución y resultados obtenidos.

1. GENERALIDADES

1.1 ANTECEDENTES

En la actualidad no solo es indispensable tener una infraestructura de TI en la que sus aplicaciones y equipos funcionen correctamente y adecuadamente, sino que también es necesario el análisis de esta información, que cada vez crece más y más abriendo una nueva era de Big Data en la que se pasa de un simple dato, al análisis de millones y millones de información, la cual permite a las compañías entender lo que sus equipos de cómputo almacenan pues en la actualidad esos datos son interpretados y analizados para la toma de decisiones futuras.

Estudiantes de la Universidad Católica De Colombia realizaron una investigación a las diferentes herramientas de autogestión y autoconfiguración existentes hoy en día una de ellas y seleccionada como la más indicada en tiempos de ejecución e interacción con los usuarios es la herramienta CHEF, afirmando que el buen uso de una herramienta de autogestión disminuye el impacto que tiene implementar una infraestructura Big Data[9].

Modelos informáticos de integración de servicios en los que la administración y configuración de equipos de cómputo toman forma cada día más en la actualidad pero también es desconocido por algunas personas relacionadas a TI, las herramientas de autogestión e integración de servicios nos permiten adoptar estos modelos informáticos para sus despliegues donde el almacenamiento distribuido es una ventaja a la hora de disponibilidad en los servicios, los sistemas de archivos distribuidos son un tipo de sistema de archivos en clúster que distribuye datos entre varios nodos de almacenamiento, generalmente para redundancia o rendimiento.

La parametrización de servicios sobre una herramienta de autogestión y autoconfiguración para ecosistemas de Big Data es una gran ventaja y una tendencia que puede mitigar todos los tiempos perdidos que tienen las empresas a la hora de desplegar diferentes servicios en su infraestructura TI.

[9] **Edwin Andrés Torres, William Alonso Rincón Saavedra. 2018.** Repository.ucatolica.edu.co. [En línea] 2018. [Citado el: 02 de 08 de 2018.] https://repository.ucatolica.edu.co/handle/10983/18238.

1.2 PLANTEAMIENTO DEL PROBLEMA

Se observan en la actualidad costos elevados en la infraestructura TI y pérdidas de tiempo exageradas por diferentes factores de planificación como son humanos y mecánicos, así como falta de buenas técnicas de desarrollo de las organizaciones[10];con la finalidad de mejorar la característica primordial que tiene una compañía que es su tiempo en la implementación y adecuación de herramientas para la autogestión de las mismas y servicios orientados a ecosistemas que permitan reducir esos tiempo perdidos o muertos y hacer los procesos más fáciles para las personas encargadas del área TI, llegando al punto de reducir no solo tiempo si no también los elevados costos de mantenimiento de la infraestructura.

El ingeniero Jorge Eduardo Reíta y Héctor Javier en su Monografía análisis de la implementación de redes Big Data para Colombia de la universidad distrital realizo un estudio de costos para empresas pymes de Colombia basado en los ofrecimientos actuales de empresas dedicadas a la implementación de servicios Big Data y reportando los costos elevados por el no uso de buenas prácticas falta de información del tema tanto en la academia como en la industria[11].

En base a lo anterior y con el propósito de ayudar a las organizaciones a desarrollar e implementar una estrategia viable de infraestructura de datos que responda de forma concreta y apropiada a las necesidad de los usuarios, resaltando que el uso de una herramienta de autogestión se puede ejecutar en cualquier ámbito entorno laboral donde se necesita optimizar y regular la administración de las plataformas de infraestructura de las compañías, gracias a la identificación de esta problemática se genera la siguiente pregunta ¿Qué parámetros y lineamientos se deben seguir a la hora de poner en funcionamiento servicios en alta disponibilidad orientados a ecosistemas de Big Data para las compañías?.

---

[10] **Maritza del Pilar SánchеZ, Avilio Villamiza. 2016.** Modelo de Costos de Servicios de Tecnologias de inforamcion en Instituciones de Esducacion superior. *TICAL.* [En línea] 13 de 09 de 2016. [Citado el: 04 de 04 de 2018.] https://documentos.redclara.net/bitstream/10786/1085/1/Modelo%20de%20Costos%20de%20Servicios%20de%20Tecnolog%C3%ADas%20de%20Informaci%C3%B3n%20en%20Instituciones%20de%20Educaci%C3%B3n%20Superior%20%282%29.pdf

[11] **Reíta Reyes, Jorge Eduardo y Salinas Hernández, Héctor Javier. 2016.** repository.udistrital.edu.co. [En línea] 2016 de 08 de 2016. [Citado el: 7 de 7 de 2018.] http://repository.udistrital.edu.co/bitstream/11349/4018/1/Big-data-FINAL-SI-1-1%20%281%29.pdf

1.3 PREGUNTA GENERADORA

¿Es útil usar una de herramientas de autogestión y autoconfiguración para desplegar servicios en alta disponibilidad para ecosistemas de Big Data?

1.4 DELIMITACION

1.4.1 Alcance:

- Definir la herramienta para la autogestión y autoconfiguración de servicios en alta disponibilidad para ecosistemas de Big Data.
- Seleccionar 4 servicios para la parametrización en la herramienta de autogestión previamente seleccionada.
- Implementar los servicios previamente elegidos, aplicados a ecosistemas de Big Data en una alta disponibilidad.
- Elaboración de manuales de configuración y de implementación para los usuarios con la finalidad de darles a ellos una guía parametrizada del montaje de los servicio y que sean usados para proyectos más adelante.

1.4.2 El espacio: El desarrollo del proyecto tendrá como ubicación los laboratorios de la facultad de ingeniería de la Universidad Católica de Colombia en Bogotá Dc así como el uso de su infraestructura TI y equipos de las salas en caso de requerirlos previamente con autorización de las personas encargadas de los laboratorios.

1.4.3 Tiempo: El Proyecto se desarrollara durante el segundo semestres del año 2018.

1.4.4 Contenido: Él trabajo de investigación tendrá toda la información necesaria para poder desarrollar el proyecto y plasmar una idea clara al lector, así como el desarrollo del mismo, adicionando figuras, imágenes y gráficas para su correcto funcionamiento.

1.4.5 Recursos: De acuerdo a las necesidades de parametrización de la herramienta de autogestión y el despliegue de un clúster se utilizaran unos equipos que cumple con las especificaciones mínimas para la correcta ejecución de los diferentes softwares a implementar.

1.4.6 Limitaciones: para el desarrollo del proyecto nos encontramos con dos limitantes muy fuertes que son el tiempo de desarrollo del proyecto y los recursos tecnológicos que necesitamos para el mismo.

2. OBJETIVOS DEL PROYECTO

2.1 OBJETIVO GENERAL

Parametrizar 4 servicios en alta disponibilidad aplicados a ecosistemas de Big Data utilizando una herramienta de autogestión y autoconfiguración.

2.2 OBJETIVOS ESPECÍFICOS

- Elaborar un estado del arte sobre las herramientas de autogestión y autoconfiguración para servicios en alta disponibilidad inmersos en ecosistemas de Big Data.
- Seleccionar 4 servicios en alta disponibilidad para la parametrización y documentación en ecosistemas de Big Data.
- Analizar el estilo de arquitectura de Big Data en alta disponibilidad para los servicios a parametrizar.
- Implementar los servicios en alta disponibilidad para el posterior despliegue desde la herramienta de autogestión.
- Realizar pruebas de rendimiento y despliegue de los servicios sobre la herramienta de autogestión

3. MARCO DE REFERENCIA

En el marco de referencia encontraremos términos y definiciones que nos permiten entender el desarrollo del proyecto así como investigaciones realizada por diferentes autores que complementaran la estructura de investigación.

3.1 ESTADO DEL ARTE

En la actualidad el uso de herramientas de autogestión y autoconfiguración en el área de TI está cogiendo fuerza cada día más a pesar que no es muy reconocido por los administradores de TI el cual se puede llegar a desplegar una gran cantidad de servicios que en la actualidad requeriría de una gran cantidad de Horas hombre para todos estos despliegue.

Cada día los clientes quieren tener más y más control de su área informática, así como el que ritmo de los negocios aumenta a diario al igual que las demandas impuestas a los equipos de aplicaciones, Lo que distingue a las organizaciones altamente competitivas es su capacidad para enviar ideas de manera rápida y exitosa[12].

Chef es una de las compañías que tiene como misión ayudar a las empresas para convertirse en organizaciones rápidas, eficientes e innovadoras impulsadas por su software[13].

Permitiendo reducir tiempos de comercialización, pobre experiencia del usuario y vulnerabilidad por falta de automatización, chef también nos proporciona otras funciones y recursos que pueden encajar en un ambiente laboral robusto y que requiera tener lo mejor de lo mejor, chef cuenta con un servidor que está disponible para cualquier versión de Linux permitiendo ser una herramienta con mucho material para entender y profundizar aprovechando sus tres garantías importantes que contiene el server de chef como lo es su instalación; la cual no representaría una molestia o una dificultad para instalarla por el profesional de TI puesto que en ella ya está previamente incluido todo lo que necesita a la hora de instalar, tender la posibilidad que acceder a información por más de 10 mil nodos, esto representa una cifra bastante elevada para acceder y administrar datos.

[12] CHEF. ¿Por qué Chef y automatización continua? [En línea] chef, 09 de 09 de 2017. [Citado el: 09 de 09 de 2017.] https://www.chef.io/why-chef/.
[13] Ibid, Sobre nosotros. [En línea] 09 de 09 de 2017. [Citado el: 09 de 09 de 2017.] https://www.chef.io/about/.

Este software está diseñado para configuration management escrito en los lenguajes de programación Erlang el cual es un lenguaje de alto nivel diseñado para escribir aplicaciones concurrentes y distribuidas[14], y el lenguaje de programación Ruby creado por Yukihiro “Matz” que gracias a la mezcla de una serie de lenguajes dio origen al Ruby baso en la idea de ser un lenguaje más poderoso que Perl, modificarlo y distribuirlo[15].

Gracias a la flexibilidad que permite el lenguaje Ruby de ser alterado por sus usuarios lo hace un lenguaje de programación amigable que no restringe a un desarrollador a la hora de plasmas sus ideas en el código A diferencia de otros lenguajes de programación orientada a objetos, Ruby se caracteriza por su intencional herencia simple. Sin embargo, Ruby incorpora el concepto de módulos (llamados categorías en Objective-C), que son colecciones de métodos.

Permitiendo así que chef use Ruby como su lenguaje de partida para contextualizar y definir todos los patrones que se encuentran implícitos en sus recursos y recetas, para así poder usar sus patrones de configuración, implementación y gestión de nodos sobre una infraestructura, gracias al poder y a la completitud que Ruby ofrece diferentes herramientas de autogestión y autoconfiguración para las áreas de TI como también lo es:

- Puppet : Herramienta de autogestión y autoconfiguración construida para multiplataforma basada en código Ruby al igual que el software chef, el lenguaje Puppet es un lenguaje declarativo usando una arquitectura cliente servidor modelado en dos capas una capa de configuración y una capa de abstracción permitiendo la configuración en cualquier tipo de sistema operativos, también cuenta con unos módulos populares para el desarrollo de cualquier infraestructura de TI (Puppet Forge, Beaker, Facter, Hiera, MCollective,Puppet DB, Razor).

- Ansible: Creado por Michael DeHann es una Plataforma que automatiza la instalación en multi-nodos destinados a través de seguridad SSH de computadoras, permitiendo el despliegue de servicios así como la compatibilidad entre diferentes sistemas operativos excepto sistema operativos Windows usando una arquitectura cliente servidor en el que el controlador es el centro de comunicación entre los diferentes nodos ayudando con la administración centralizada de infraestructuras TI.

[14] ERLANG. erlang. [En línea] 10 de 11 de 2017. [Citado el: 10 de 11 de 2017.] http://www.erlang.org/about

[15] RUBY. Acerca de Ruby. [En línea] Ruby y jekyll. [Citado el: 07 de 10 de 2017.] https://www.ruby-lang.org/es/

- Vagrant: Es una herramienta gratuita lanzada el 8 de marzo de 2010 para crear entornos de desarrollo disponible en Windows, MacOs y GNU/Linux partiendo de compartir ficheros vagrantfile con algún otro desarrollador con el fin de usar un simple comando para reproducir el mismo entorno de desarrollo, usando Ruby como lenguaje principal pero puede ser usado en otros proyectos con lenguajes como PHP, Python Java C# y JavaScript.

- SaltStack: Software de gestión y configuración de infraestructura de código desarrollado por Thomas S Hatch escrito en pitón usando un repositorio central con el fin de automatizar todo tipo de tarea administrativa repetidas en el sistema reduciendo errores que surgen cuando organizadores de TI configuran sistemas.

Debido a que los administradores del área de TI pasan más del 50% de su tiempo en tareas rutinarias y repetitivas, la automatización del área de TI reducirá el tiempo dedica a este tipo de tareas, pero no solo cambiar de rutinas a los administradores se puede conseguir con puppet también se puede conseguir; una alta disponibilidad y confiabilidad, mejor escalabilidad, mejor adaptabilidad, una entrega de calidad en los códigos fuentes[16].

Las herramientas de autoconfiguración permiten gestionar esos entornos en lo que existen una pluralidad de plataformas incluidos Windows y Linux así como su administración y procesos manuales de trabajo que son imprescindibles para avanzar más rápido y mitigar las variables que afectan el desarrollo de los proceso en las compañías y más cuando ya se están acabando los recursos para una buena mejora en infraestructura virtualización o de cloud computing estos puede llegar a mejorar el desarrollo de cualquier infraestructura, gracias a esto Michael DeHaan autor de Ansible decide aprovechar los conocimientos que ha adquirido en el grupo de tecnologías en el que ha trabajado de Red Hat donde se busca ser una de las plataformas más proliferas a nivel mundial bajo su plataforma.

Openstack es la combinación de herramientas con código abierto que usan recursos virtuales interrelaciónalos para administras nubes tanto públicas como privadas, Seis de estos proyectos manejan los principales servicios de cómputo en la nube de computación, redes, almacenamiento, identidad y servicios de imágenes, mientras que más de una docena de proyectos opcionales se pueden agrupar para crear nubes únicas e impleméntales[17].

---

[16] PUPPET. 9 maneras en que la automatización de TI lo hace más exitoso. [En línea] [Citado el: 16 de 11 de 2017.] https://puppet.com/resources/whitepaper/9-ways-it-automation-makes-you-more-successful

[17] OPENSTACK. Software de código abierto para la creación de nubes privadas y públicas. [En línea] [Citado el: 12 de 12 de 2017.] https://www.openstack.org/home-2?utm_expid=.OxTNoTdOSrqf1ItR3pCEsw.1&utm_referrer=https%3A%2F%2Fwww.google.com.co%2F.

A pesar de la facilidad y funciones que permiten las plataformas de administración y virtualización, Openstack realmente usa los recursos virtuales para ejecutar una combinación de herramientas. Estas herramientas crean un entorno de nube que cumple con los 5 criterios del Instituto Nacional de Estándares y Tecnología de computación en la nube: una red, recursos agrupados, una interfaz de usuario, capacidades de aprovisionamiento y control / asignación automática de recursos[18].

La disciplina dedicada a procesamiento, análisis y almacenamientos de datos a gran escala es conocida como Big Data en el que el uso de herramientas y metodologías sirven para la toma de decisiones partiendo de unas series de combinaciones de negocios-analítica-digitalización-medios sociales-hiperconectividad y computación en la nube así como las características principales de Big Data de gran volumen de almacenamiento ,la velocidad en cantidad de tiempo que toma la data en ser procesada, pero una de sus fuertes es la variedad ya que es utilizable en múltiples formatos y tipos de datos dando a la empresa una utilidad en estos[19].

[18] NIST. NIST Cloud Computing Program - NCCP. [En línea] National Institute of standars and Technology U.S. [Citado el: 10 de 12 de 2017.] https://www.nist.gov/programs-projects/nist-cloud-computing-program-nccp

[19] TICXAR, SUNQU Y. Conceptos y ecosistema big data. [En línea] [Citado el: 11 de 09 de 2017.] http://www.zemsania.com/recursos-zemsania/whitepapers/DTS/Ecosistema_Big_Data.pdf.

3.2 MARCO CONCEPTUAL

3.2.1 Automatización:

Es la reorganización de la información en tiempo real que sea accesible a todo el personal involucrado en la operación; su uso en el proceso provee un conjunto de técnicas de comunicación, computación y equipamiento de oficina utilizadas con la finalidad de aumentar la productividad y calidad de la gestión de la operación.

3.2.2 Análisis de datos:

Es conocido también como una disciplina de examinar y explorar cualquier tipo de datos para encontrar hechos, relaciones, patrones y/o tendencias. El cual es realizado por personas a partir de una parametrización.

3.2.3 Business Inteligencie:

Es la habilidad que se tiene para transformar los datos en información y la información en conocimiento con el fin de maximizar cualquier tipo de riesgo que se presente, por medio de metodologías, aplicaciones y tecnologías que permitan transformar los datos e información para las compañías para su explotación directa (reporting, análisis OLTP / OLAP, alertas).

3.2.4 Bases De Datos:

Es aquel lugar en el que se puede almacenar información de una forma ordenada con diferentes propósitos y uso, dependiendo de esto se pueden usar ciertos tipos de bases de datos para que se acople a las necesidades de los que la requieran.

3.2.5 Almacenamiento distribuido:

Arquitectura en la que la información se guarda en varios servidores, lo que da como resultado mayor flexibilidad y mejor rendimiento cuando se estén manejando los datos, esto permite que la información no se pierda cuando alguno de los servidores falle actuando como si no se tuviera ninguna falla en la infraestructura por parte de al servidor abajo.

3.2.6 Alta disponibilidad:

Término que es tendencia hoy en día en el área de la informática para hacer referencia a prácticas y medidas para garantizar la efectividad de un servicio durante las 24 horas del día.

Permitiendo así a la compañía protegerse contra las pérdidas de ingresos cuando el acceso a sus centrales de información se vea interrumpida, planificándolas en 5 categorías[20]:
-paradas planificadas.
-paradas imprevistas.
-restauración ante siniestro.
-reducción de la ventana de copia de seguridad.
-distribución equilibrada de la carga.

3.2.7 Clúster:

Tipo de computación paralela o de procesamiento distribuido, formado por una colección de computadores individuales interconectado entre si trabajando conjuntamente en un objetivo unificado de computo[21].

3.2.8 Minería de datos:

Termino regularmente usado para referirse al proceso de información procesable del conjunto de grandes datos usando un patrón analítico matemático definido en un modelo aplicable a diferentes escenarios como lo son:

-pronostico.
-riesgo y probabilidad.
-recomendaciones.
-búsqueda de secuencias.
-agrupación.

[20]Alta Disponibilidad. [En línea] IBM, 17 de 09 de 2016. [Citado el: 20 de 06 de 2018.] https://www.ibm.com/developerworks/ssa/library/bd-archpatterns1/index.html.

[21] **NAVAS, ANTONIO ANTIÑOLO.** Clusters. [En línea] [Citado el: 02 de 03 de 2018.] http://www.inf-cr.uclm.es/www/sbenito/AIC/Cursos%20Anteriores/Curso%202004-2005/Transparencias/Clusters.pdf.

3.2.9 Ingestión:

Son Tecnologías orientadas a la recolección de los datos desde su origen, por ejemplo en bases de datos tradicionales, o en flujos continuos a través de la red.

3.2.10 Mensajería:

Tecnologías que permiten el intercambio de datos entre los diferentes componentes software de manera eficiente, en esta categoría se ubican todas las colas de mensajes.

4. METODOLOGÍA

4.1 DESCRIPCIÓN GENERAL

La ejecución y desarrollo del proyecto seguirá los lineamientos basados en el método científico con una estructura de 5 pasos enumerados a continuación.

1. Observación y análisis de una herramienta de autogestión y autoconfiguración.

2. Hipótesis de selección para la parametrización de la herramienta de autogestión en ecosistemas de Big Data.

3. Experimentación y diseño de la herramienta de autogestión y autoconfiguración para ecosistemas de Big Data

4. Teorías de confirmación e implementación sobre una herramienta de auto gestión.

5. Ley demostrativa con pruebas y resultados obtenidos sobre el desarrollo de objetivos del proyecto

Figura 1. Etapas del método científico

Fuente: ELESAPIENS**.** El metodo Cientifico. [En línea] LEARNIN & FUN, 06 de 02 de 2018. [Citado el: 06 de 02 de 2018.]

4.2 ETAPA1

Observación y análisis de una herramienta de autogestión y autoconfiguración.

Se observara y se analizará la herramienta Chef centrada en la autogestión y la autoconfiguración que permita enlazar y desplegar 4 servicios en alta disponibilidad para ecosistemas de Big Data que son Apache haddop, Apache cassandra, spark y MongoDB todos con su última versión.

4.3 ETAPA 2

Hipótesis de selección para la parametrización de la herramienta de autogestión y autoconfiguración en ecosistemas de Big Data.

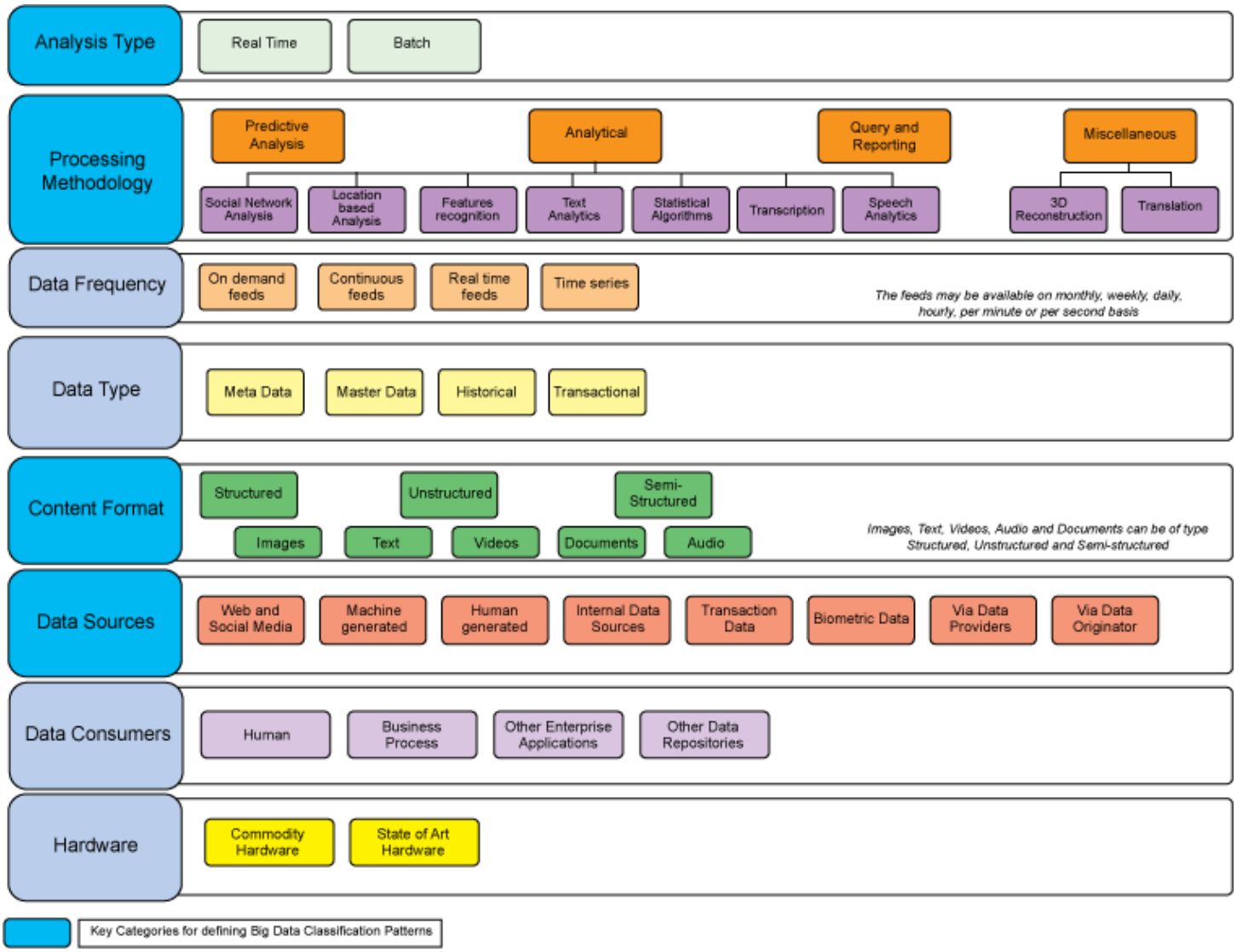
Partiendo de una hipótesis en la que las direcciones de TI tienen aún dificultades en el levantamiento de servicios por su elevado costo y en ocasiones por la falta de conocimientos de sus encargados se pierde algo muy importante para las compañías que es el tiempo y el análisis de sus datos.
Con esta hipótesis levantaremos gran información que nos permitirá hacer un pre alistamiento de los parámetros que más se acomodan en la herramienta, para pasar a la siguiente etapa de diseño.

4.4 ETAPA 3

Experimentación y diseño de la herramienta de autogestión y autoconfiguración para ecosistemas de Big Data.

Con los servicios parametrizados y la alta disponibilidad de los mismos se procede experimentalmente al diseño que tendrá la herramienta de autoconfiguración a la hora de exponer sus servicios teniendo claro diferentes patrones y métricas de información siguiendo la estructura que tiene los ecosistemas de Big Data y sus categorías.

Figura 2.Clasificación de arquitectura Big Data.



FUENTE: Clasificación de Big Data. [En línea] IBM, 17 de 09 de 2016. [Citado el: 20 de 06 de 2018.] https://www.ibm.com/developerworks/ssa/library/bd-archpatterns1/index.html.

4.5 ETAPA 4

Teorías de confirmación e implementación sobre una herramienta de autogestión.

En esta etapa de ejecución del método científico ponemos en marcha los servicios en alta disponibilidad sobre nuestra herramienta de autogestión con las

configuraciones previamente estudiadas y analizadas durante las etapas anteriores con la finalidad de ir realizando manuales de usuario.

4.6 ETAPA 5

Ley demostrativa con pruebas y resultados obtenidos sobre el desarrollo del objetivo del proyecto.

Una vez ya concluido con la parametrización de los servicios en alta disponibilidad para ecosistemas de Big Data sobre la herramienta de autogestión y autoconfiguración, se generaran rutinas para el análisis de información en cada uno de los servicios, que se desplegaron para analizar tiempos y recursos comparados con procesos actuales de TI y a su vez demostrar con pruebas la viabilidad del proyecto.

5. DESARROLLO DEL PROYECTO

5.1 ETAPA1

En nuestra primera etapa de desarrollo del proyecto usamos la herramienta de autogestión y auto configuración Chef para dar continuidad con la investigación realizada el trabajo de grado de los estudiantes Edwin Andrés Torres Robles y William Alonso Rincón Saavedra en la implementación de una herramienta de autogestión y auto configuran, gracias al estudio y a las pruebas obtenidas en el desarrollo de ellos, nos basamos para darle continuidad al uso de la herramienta de tal manera que se redujera tiempo en el desarrollo de investigación de la mismas, basados en el propósito y la orientación del proyecto de grado, línea investigativa de ecosistemas Big Data, Seleccionamos 4 servicios por vocación y por experiencia en el manejo de los mismo.

- Apache Hadoop.
- MongoDB.
- Apache Cassandra.
- Apache Spark.

Estos servicios en la siguiente etapa de desarrollo serán configurados para que sean ejecutados en un ambiente de alta disponibilidad, aunque la estructura de algunos de estos servicios cuenten con un modelo arquitectónico distribuido de datos no asegura que sea en alta disponibilidad.

5.2 ETAPA2

La hipótesis de selección para la parametrización de la herramienta de autogestión y autoconfiguración en ecosistemas de Big Data es el déficits que tienen las áreas de TI, para que el levantamiento de servicios se concluye en tres hipótesis que surgieron en el estado del arte mencionados a continuación.

5.2.1 Reducción de costos en el desarrollo de plataformas TI:

El factor dinero en una compañía es la clave de muchos existes así como el buen manejo de los mismo, darle un uso adecuado a los recursos de la compañía permite que esta tenga un crecimiento exponencial bastante elevando de su economía, por ende trabajamos sobre herramientas open source las cuales están codificadas de forma free para reducir esos temas de dinero en licencias y sistemas operativos con grandes costos como lo es sistemas Windows para el cual tendríamos que tener US 6.155 dólares para tener una edición de Windows server 2016 Datacenter a usar un sistema Linux totalmente gratis sin discriminar componentes adicionales que elevarían el costo de plataformas TI

5.2.2 Recursividad en el manejo de scripts:

Con el desarrollo del proyecto buscamos diseñar unos scripts que fueran acordes a la necesidad de cada estilo de arquitectura sin dejar a un lado sus aspectos importantes y los cuales lo hacen ideal para ser utilizados en áreas de la tecnología como son servicios de Big Data y que sean servicios desplegados en alta disponibilidad manejando la estructura inicial que la herramienta de autogestión y auto configuración nos brinda, basados en el lenguaje de programación Ruby en el cual vamos a recolectar todos los recursos de configuración necesarios y primordiales para el correcto despliegue del mismo.

La ejecución de las recetas será por parte de un cliente-chef que se iniciara cuando el mismo la solicite sin tener que estar modificando o creando nuevas recetas para su uso posterior, esta receta estará alojada en el chef-server hasta que el administrador del mismo lo permita y las modificaciones que se le realicen no afectaran a los clientes que ya las utilizan solo serán afectados los nuevos nodos que requieran de ella.

5.2.3 Tiempos muertos en intervalos de instalación:

Una problemática que se tiene a la hora de desplegar servicios sin usar un agente o herramienta de autogestión y autoconfiguración es que se tienen que iniciar de cero cada vez que se requiera levantar un servicio en un nodo u o equipo diferente, es por esto que una hipótesis son los intervalos muertos de tiempo en instalaciones, con la herramienta de autogestión y autoconfiguración podemos almacenar estar recetas en un libro de recetas en el que dependiendo de su arquitectura serán descargadas y ellas de forma automática se configuraran el cliente-chef.

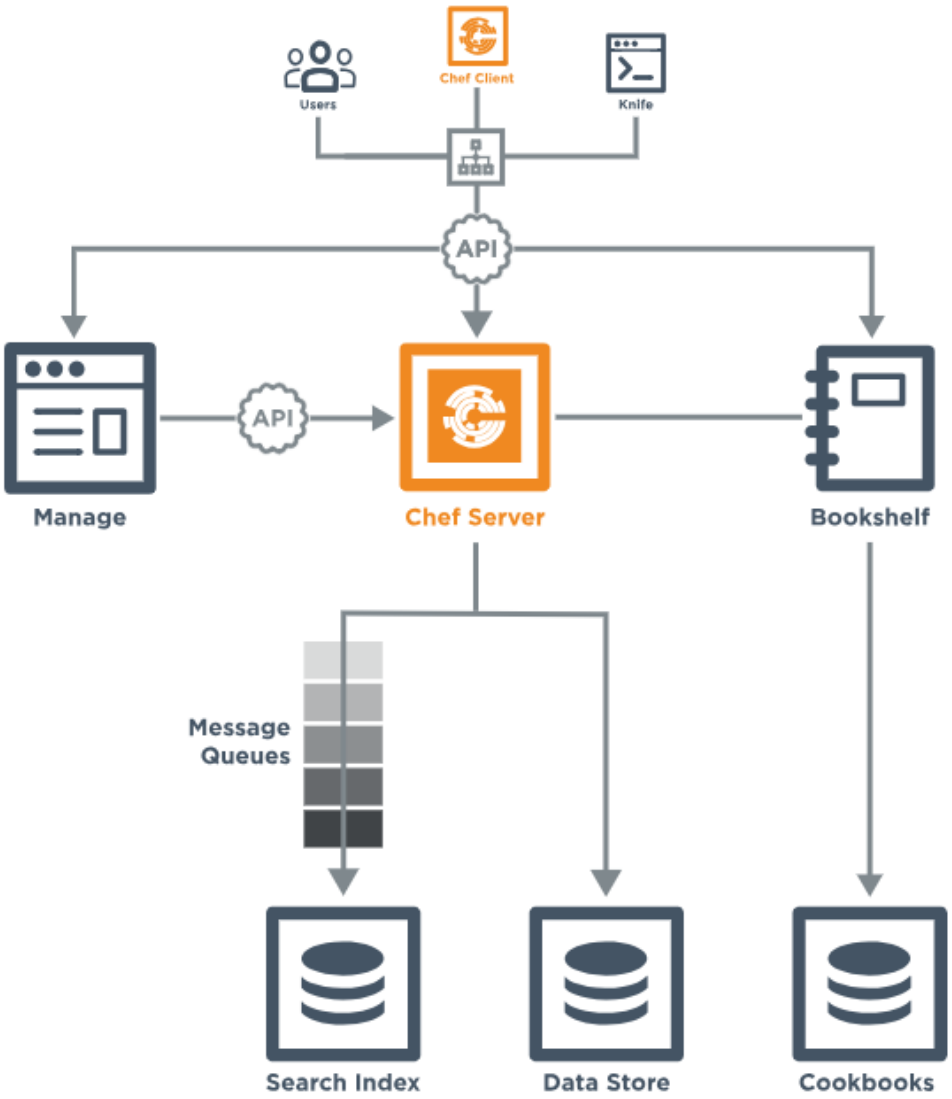
5.3 ETAPA 3

En la experimentación y diseño de la herramienta de autogestión y autoconfiguración para ecosistemas de Big Data se partió de instalar de forma ordena y como lo explican los desarrolladores de cada servicio para tener un script de cada servicio en el que se evalúa tiempo y percances que se tiene en el montaje antes de usar la herramienta de despliegue para este tipo de servicios.

Una vez levantados los 4 servicios en alta disponibilidad para ecosistemas de Big Data, se procede a instalar CHEF la cual es la herramienta de auto gestión y auto configuración que se seleccionó previamente para este proyecto y sobre la cual se desarrollara todo el proyecto, resaltando que CHEF tiene tres componentes para su correcto desarrollo y configuración, que sin la implementación de alguno de ellos no se podría llevar a satisfacción su despliegue.

- **chef server:** El chef server es un componente principal de CHEF que actúa como un cubo en el que se almacenan los libros de recetas, las políticas que regirán a los nodos y sus metadatos que contiene a cada nodo registrado, en el chef server se reciben las peticiones de cada cliente chef sobre el requerimiento de alguna receta en específica y su configuración esta predeterminada en la ruta /etc/opsconde/chefserve.rb así como sus componentes:

Figura 3. Componentes del servidor

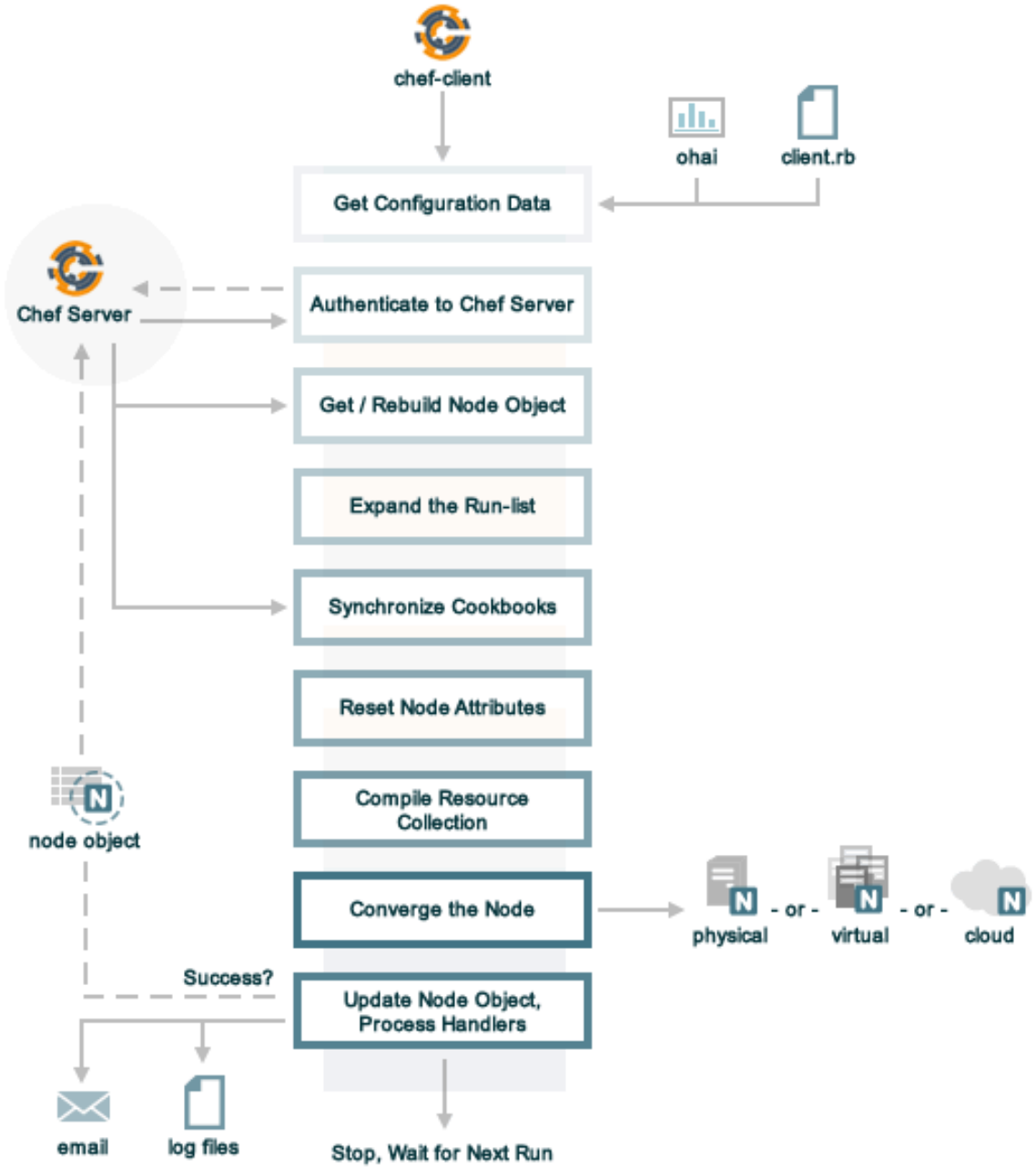


FUENTE: server_components. [En línea] 26 de 07 de 2016. [Citado el: 12 de 08 de 2018.] https://docs.chef.io/_images/server_components.svg

- **chef Workstation:** las estaciones de chef permiten crear y manipular los libros de recetan que se usaran el en el chef server, ya que chef corre basado en el modelo de arquitectura cliente servidor se debe configurar una estación de trabajo que pueda interactuar con el servidor más a delante permitiendo así una ejecución remota, análisis y configuración de tareas, herramientas para la creación de libros de recetas todo esto está inmerso en el paquete de instalación dependiendo del sistema operativo para nuestro proyecto es centos 7.

- **Chef cliente:** Es el representante de cada nodo que está bajo la supervisión principal de chef, cada chef clientes debe tener unas características especiales para poder llevar los nodos al estado deseado a lo que se conoce como el chef cliente run

Figura 4. Componentes de Chef-client



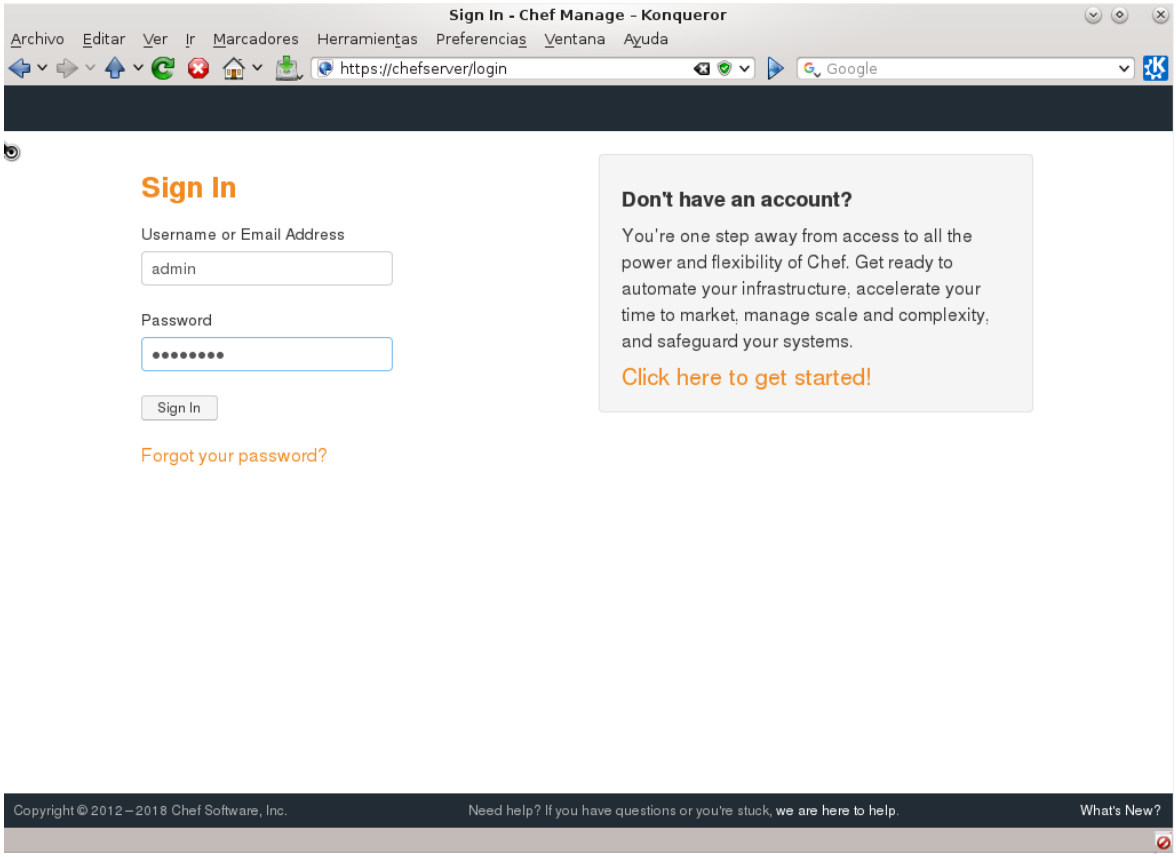
FUENTE: Chef-cliente. [En línea] 02 de 10 de 2017. [Citado el: 25 de 09 de 2018.] https://docs.chef.io/_images/chef_run.png.

5.4 ETAPA 4

Después del levantamiento de los tres componentes de CHEF se procede a registrar los nodos que usaremos así como las cuatro recetas de los de los servicios previamente configuradas para su despliegue en la herramienta de autogestión

Ingresando a nuestro servidor desde cual quiera de los 3 componentes de chef bien sea el servidor la estación de trabajo o el cliente podemos ver que ya nuestro chef manager se encuentra arriba eso significa que nuestro CHEF fue instalado y configurado correctamente, las claves de ingreso serán configuradas por el desarrollador o si por defecto se desean usar las que trae el paquete de instalación de chef. VER *manual de configuración e instalación chef server*

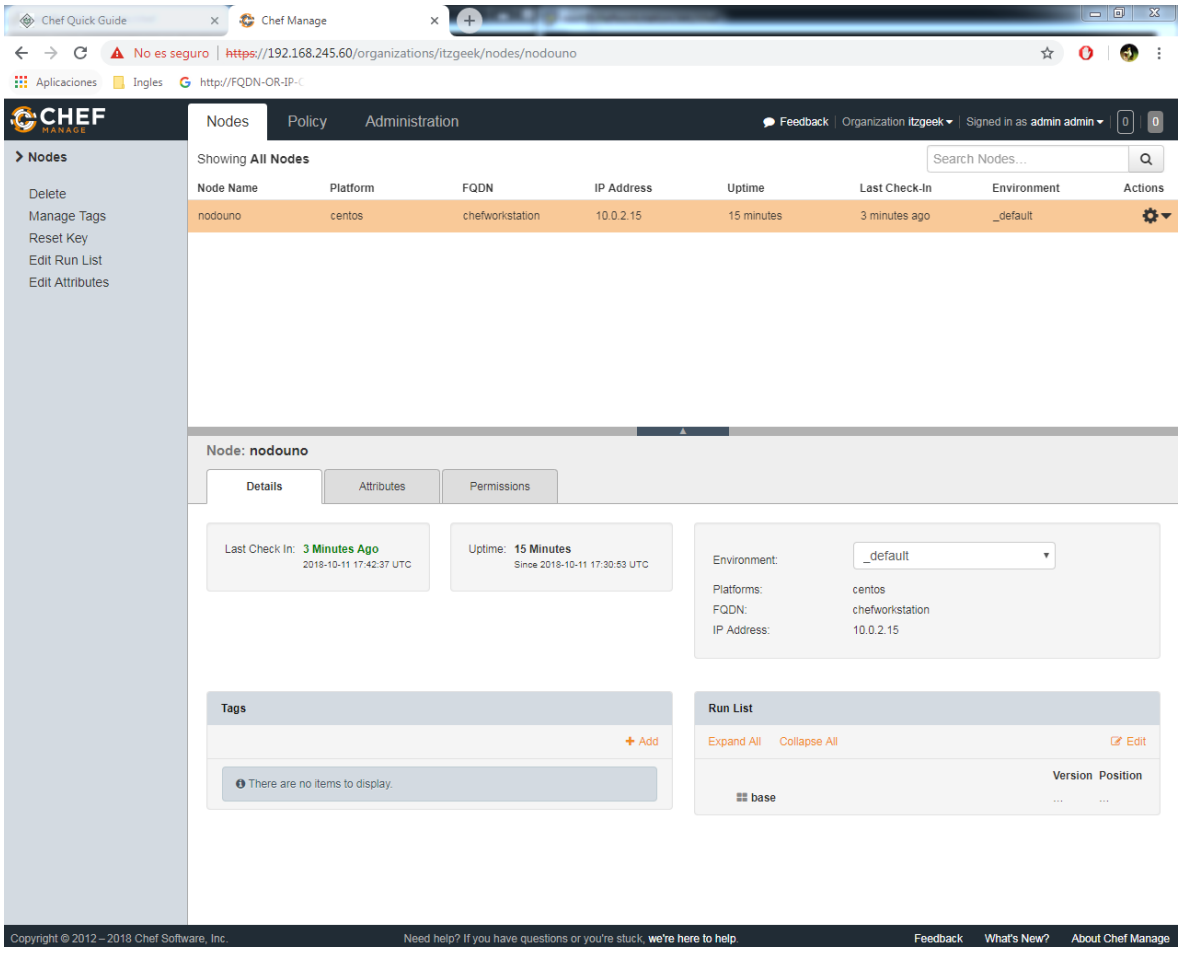
Figura 5. Chef manager up



FUENTE: el autor

Una vez logeados en nuestra consola de administración nos aparecerá el módulo de nodos en el que tendremos que definir el número de nodos que se necesitaran dependiendo de la topología de nuestra arquitectura de software, así como las políticas de administración de nuestro software chef, las configuraciones para un ecosistema de Big Data en alta disponibilidad y las recetas previamente cargadas al server, se pueden VER *manual de configuración de chef*

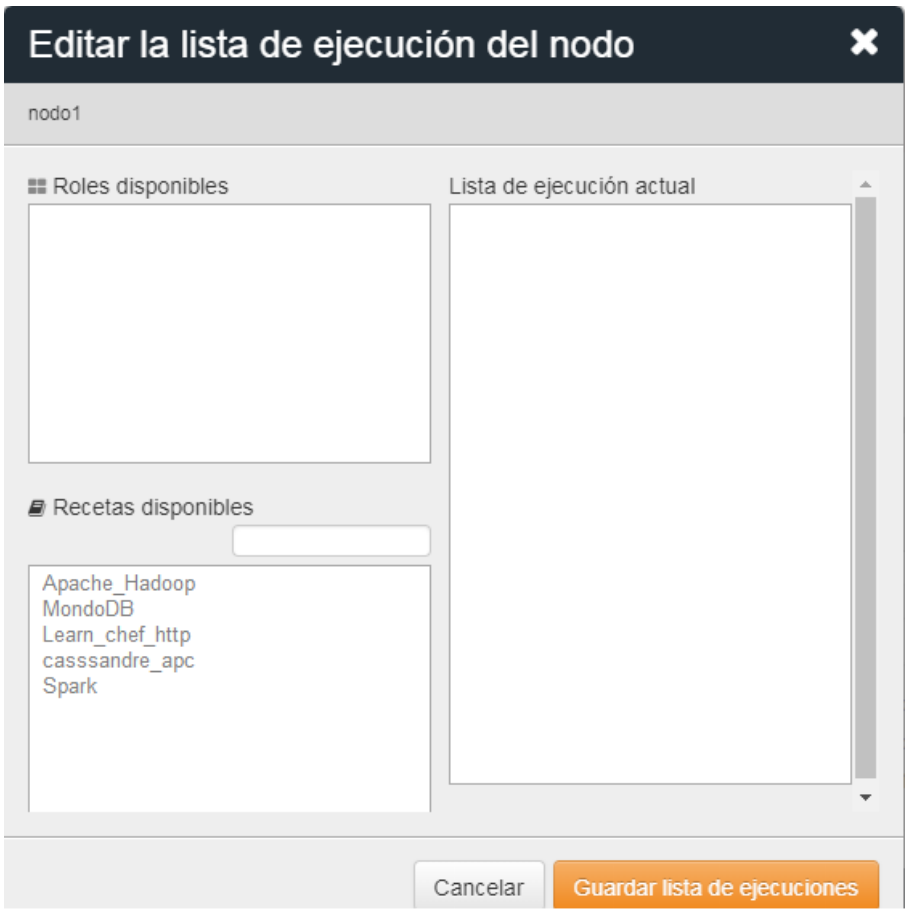
Figura 6. Centro de registro de nodos



FUENTE: el autor

Una de las ventajas y la utilidad que tiene la herramienta de auto gestión y auto configuración es que se puede parametrizar las recetas disponibles para cada nodo dependiendo el número de nodos que la organización requiera aclarando que estas recetas son subidas exclusivamente por el Workstation que es el componente encargado de realizar, modificar y eliminar una rutina para nuestro proyecto seleccionamos solo 4 servicios pero chef puede manejar indeterminados servicios depende más del equipos de hardware en el que se tenga alojando todo que del propio chef VER *manual de configuración e instalación chef, apaché hadoop, mongDB, cassandra, spark*.

Figura 7. Lista de nodos por ejecutar

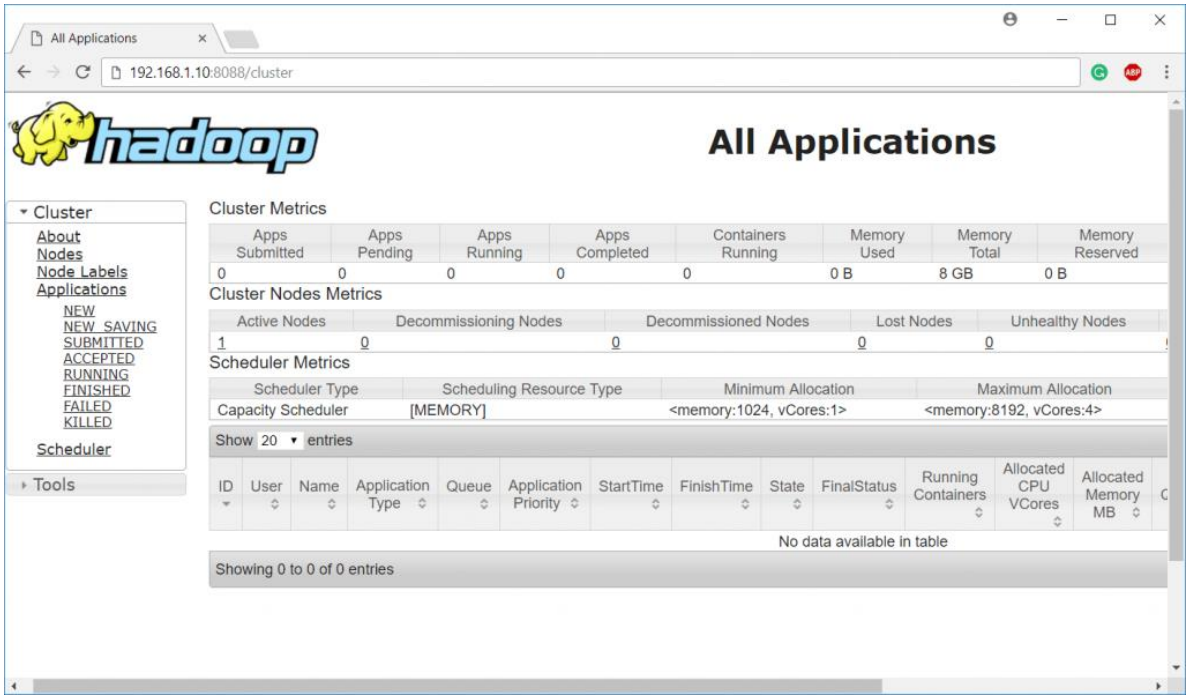


FUENTE: el autor

5.5 ETAPA 5

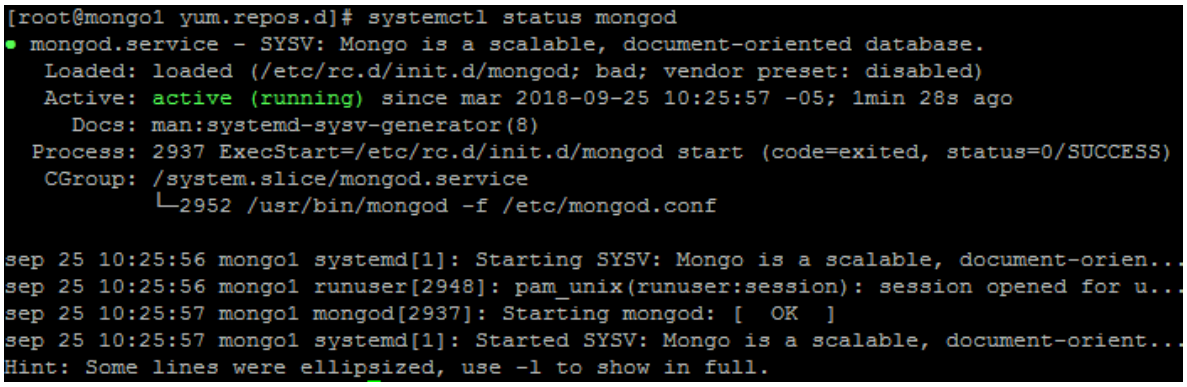
En el etapa 5 y ultima de nuestra metodología de desarrollo de proyecto tenemos que principalmente se levantaron 4 servicios para ecosistemas Big Data luego se levantó el software chef con sus 3 componentes y una vez teniendo estos 3 componentes arriba se procedió a crear las recetas a partir de las rutinas de las cuatro servicios anterior mente descritos y con la configuración de clúster para que fueran servicios en alta disponibilidad uno de los objetivos de la investigación.

Figura 8. Instalación de Hadoop exitosa.



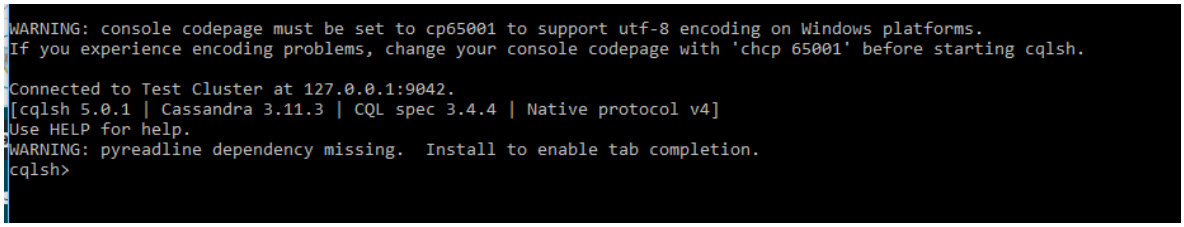
FUENTE: el autor

Figura 9. Instalación de MongoDB exitosa.



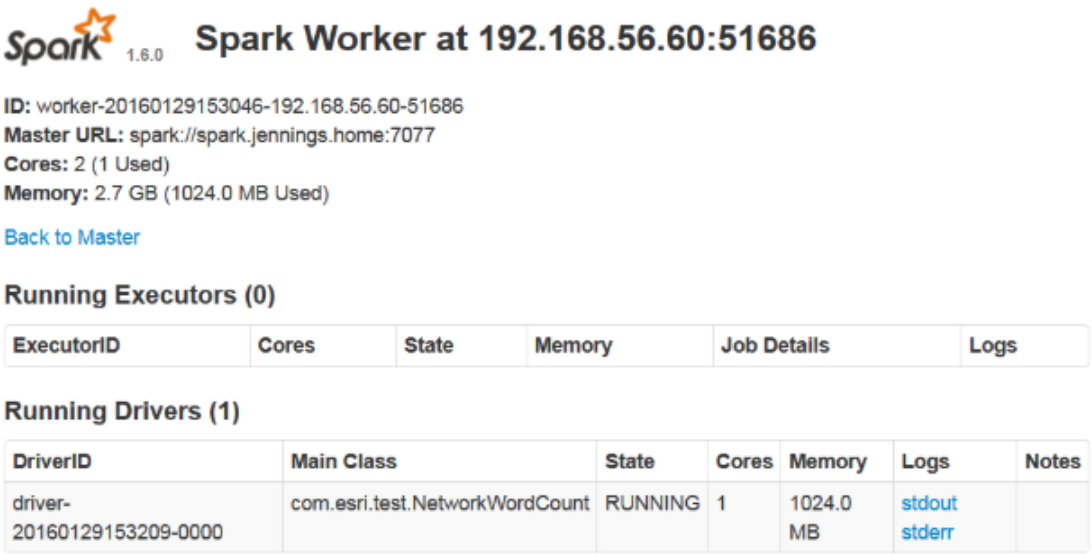
Fuente: el autor

Figura 10. Instalación de Cassandra exitosa.



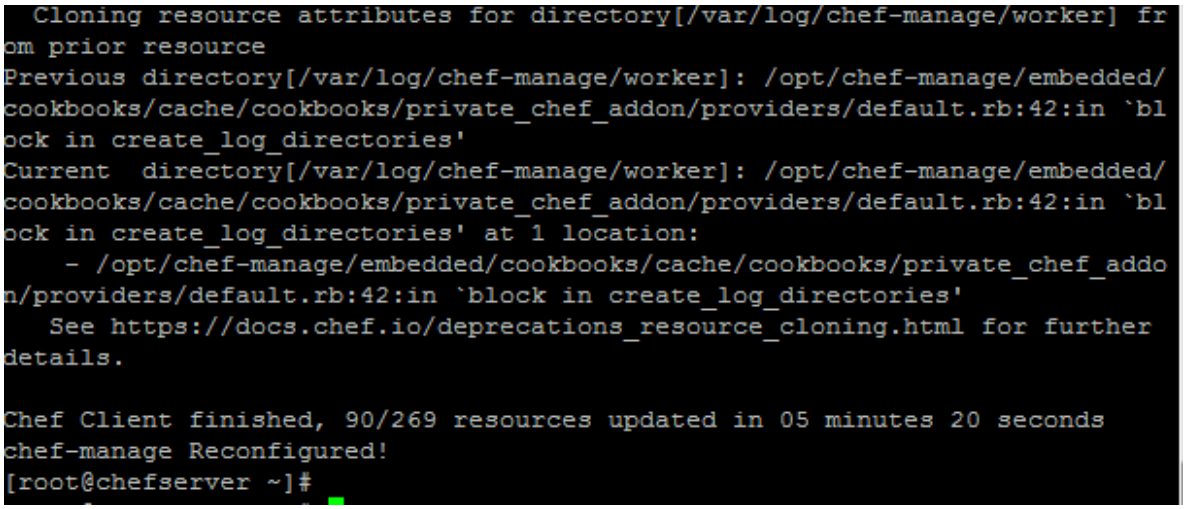
Fuente: el autor

Figura 11. Instalación de apache spark.



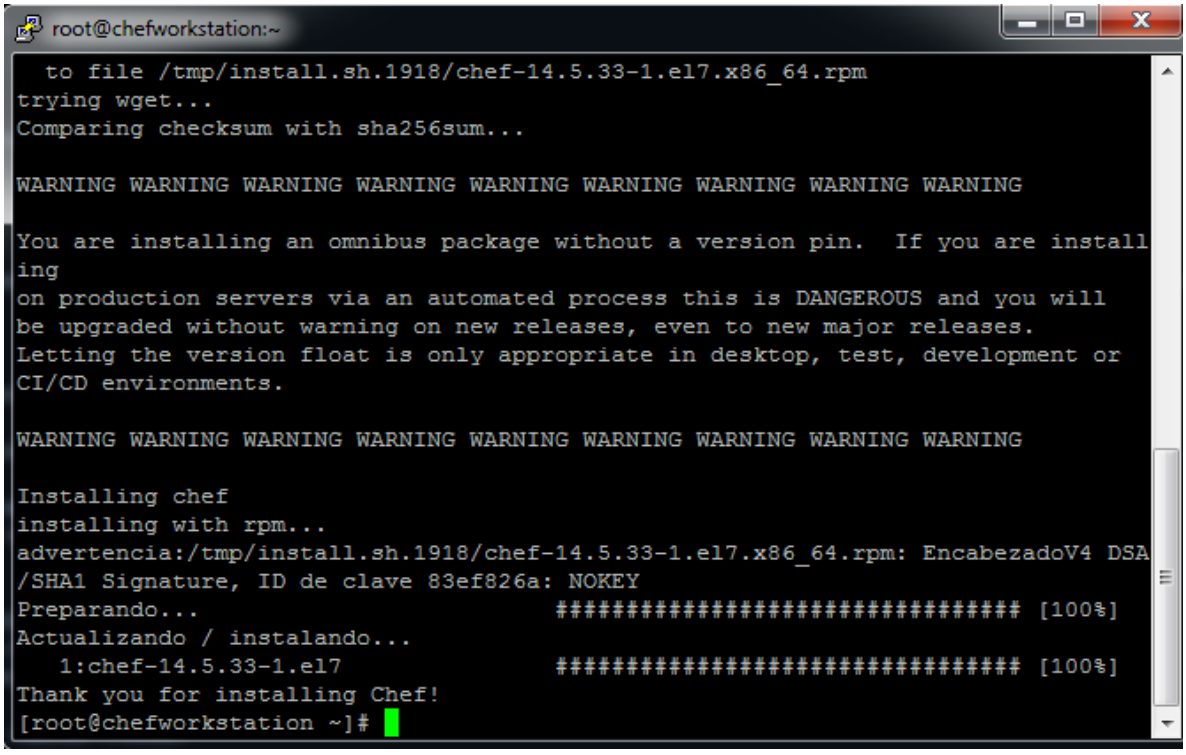
Fuente: el autor

Figura 12. Instalación de Chef manager.



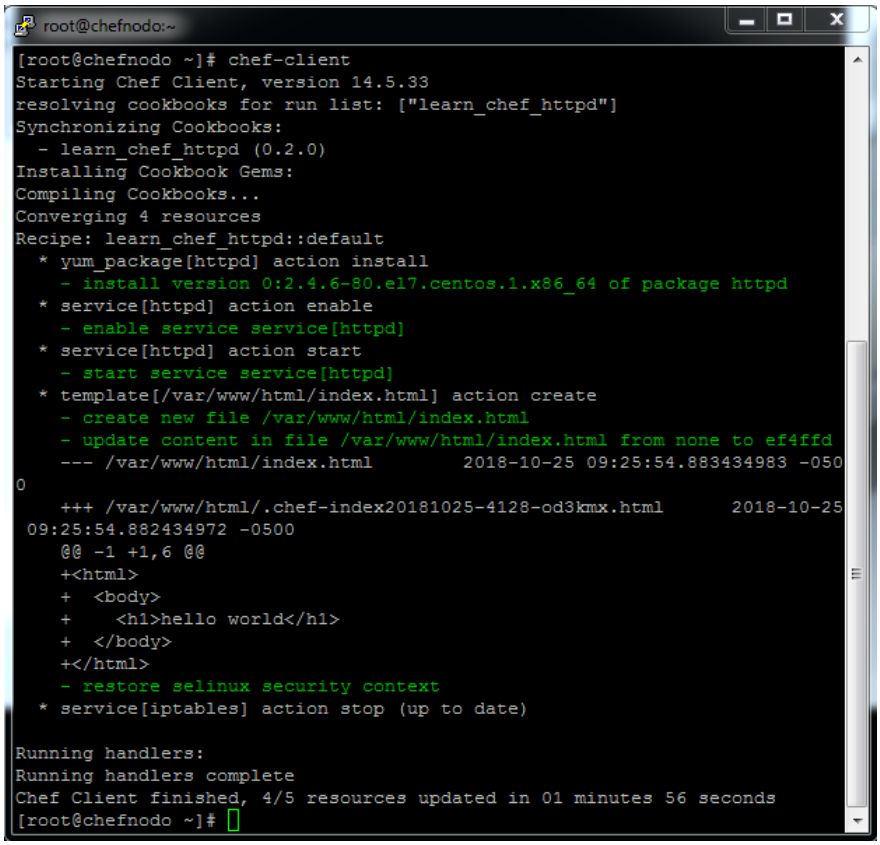
Fuente: el autor

Figura 13. Instalación de chef Workstation.



Fuente: el autor

Figura 14. Instalación de chef-cliente.



Fuente: el autor

Creamos un libro de cocina que llamaremos método el cual tendrá nuestras 4 recetas principales

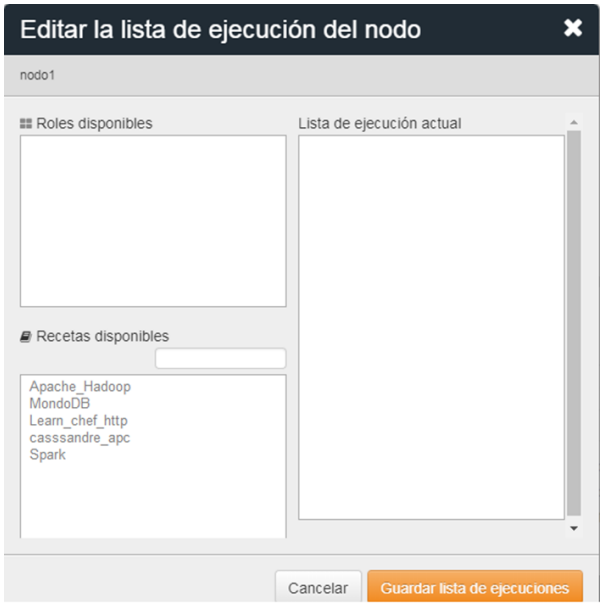
Figura 15. Instalación de un libro de cocina.

```
#
# Cookbook Name:: motd
# Recipe:: default
#
# Copyright 2018, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#
#file '/etc/motd' do
#       content 'welcome to chef'
#end
```

Fuente: el autor

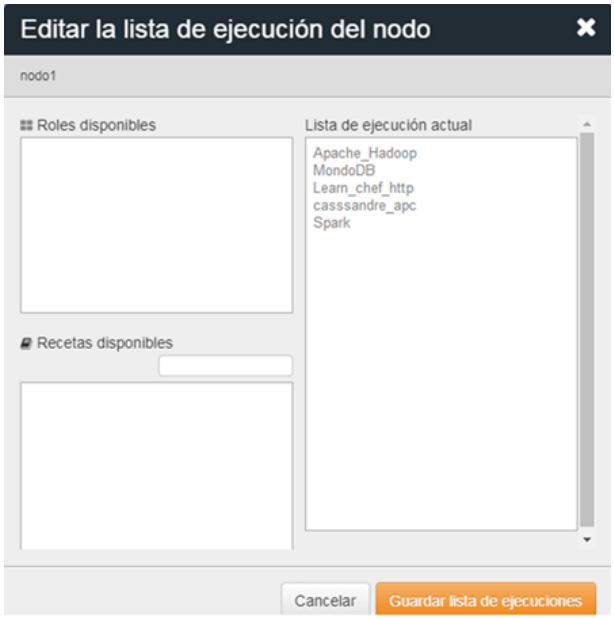
Procedemos a cargar las recetas de los cuatro servicios (VER: manual de instalación y configuración del software chef) en el libro de cocina y verificamos que las recetas estén disponibles.

Figura 16. Recetas cargadas a chef



FUENTE: el autor

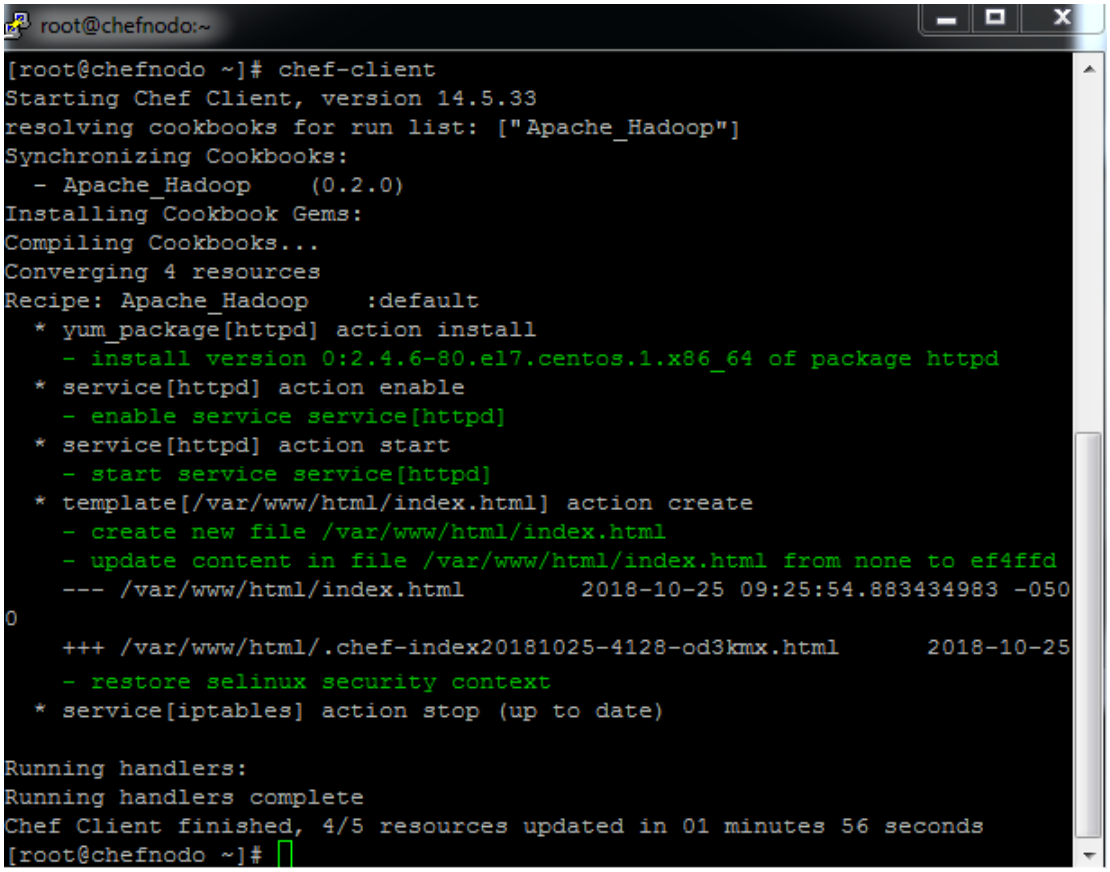
Figura 17. Lista de ejecución de recetas.



FUENTE: el auto

Finalmente nuestros servicios fueron agregados a la lista de ejecución de la consola administrativa de chef resta ejecutar la receta en el nodo a utilizar (VER: manual de instalación y configuración del software chef) con solo ejecutar un nodo ya la receta se replica para los demás nodos que conforman el nodo una ventaja enorme si se tiene en cuenta que pueden ser varios nodos en vez de uno.

Figura 18. Despliegue de servicios.

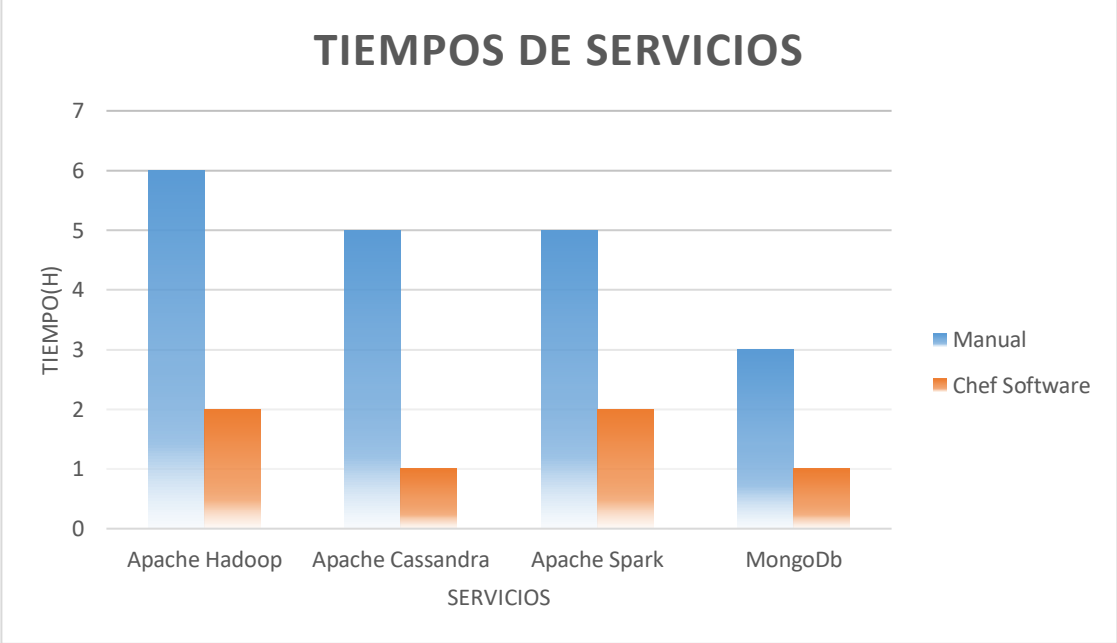


Fuente: el autor

En nuestra última etapa de desarrollo de nuestra metodología logramos levantar los servicios en alta disponibilidad gracias al uso de la herramienta de autogestión y autoconfiguración en la que parametrizada las estructuras de los servicios logramos minimizar tiempos a un 60% si se desplegaran de forma manual teniendo en cuenta que el desarrollo fue para un clúster de máximo 4 nodos por servicio en el que Chef lo asume como una sola receta para los cuatro nodos usando 120 líneas de código Ruby para Hadoop, 110 para Casandra 112 para apache spark y 80 para MongoDB

si se realizara de forma manual sin el uso se la herramienta seria el mismo número de líneas de código multiplicadas por el número de nodos que conformarían el clúster en alta disponibilidad que una de las ventajas que se puede obtener a la hora de usa Chef, en la figura 19 se ilustra los tiempos que lleva desplegar los servicios en alta disponibilidad usando la herramienta de autogestión y usando la forma tradicional.

Figura 19. Tiempos de ejecución.



Fuente: El Autor

6. ENTREGABLES DEL PROYECTO

6.1 MANUAL DE PAREMETRIZACION DE SERVICION EN ALTA DISPONIBILIDAD PARA ECOSISTEMAS DE BIGDATA

- Documento implementación de la parametrización de servicios en alta disponibilidad para ecosistemas de Big Data.

6.2 MANUALES DE INSTALACION Y CONFIGURACION

- Manuales de instalación y configuración en alta disponibilidad de los 4 servicios implementado.
- Manual de instalación y configuración del software chef.

6.3 SCRIPTS DE LAS RECETAS IMPLEMENTADAS Y DE LOS SERVICION IMPLEMENTADOS

- Scripts de programación usado para las recetas de los 4 servicios implementados en la herramienta de autogestión.
- Scripts de programación usado para levantar los 4 servicios sin el uso de la herramienta de autogestión.

7. CONCLUSIONES

Con el desarrollo de esta investigación, se comprueba que las herramientas de auto gestión y autoconfiguración son de gran ayuda para las áreas de TI reduciendo mucho los procesos reiterativos a la hora de desplegar servicios permitiendo que la las compañías tengan una mejor parametrización a la hora de implementar algún tipo de arquitectura que requiera el consumo de servicios en alta disponibilidad, ya que no tienen que estar instalando un servicio cada vez que lo requieran sí no que estarán almacenados en un server que les permitirá a ellos usarlos las veces que lo requieran solo con una configuración inicial.

El uso de servicios en alta disponibilidad le da un valor agregado a la compañía que lo implemente en su área de TI ya que gracias a esta buena práctica sus servicios siempre estarán disponibles aunque alguno de los nodos caiga o su estado sea detenido de forma no programada permitiendo así que los nodos esclavos suban sin que el usuario se percate del mismo y su información sea protegida.

Reduce entre un 55% y 60% el tiempo de ejecución de un servicio en alta disponibilidad mitigando fallos en instalaciones futuras ya que no es necesario modificar las recetas a la hora de una nueva instalación, solo se debe modificar dicha receta si la arquitectura de TI cambia.

8. RECOMENDACIONES

Es importante tener claro un concepto básico de los lenguajes de programación a utilizar para el uso de la herramienta así como tener definidos los tipos de servicios a utilizar para que sean de una forma amigable con la compañía o con el personal que desee implementar los recursos.

Es fundamental contar con equipos de hardware óptimos y en completo funcionamiento para que el desarrollo de las prácticas se haga de manera exitosa.

9. TRABAJOS FUTUROS

El anterior proyecto abre campo para todas aquellas personas que deseen implementar la Herramienta de autogestión y autoconfiguración en un ambiente real como lo seria las salas de cómputo de la Universidad Católica De Colombia en el que la instalación de algún componente es de forma manual, a su vez se puede usar este proyecto como guía para la implementación de servicios en alta disponibilidad en la nube usando la herramienta de autogestión

10. BIBLIOGRAFÍA

Atributos. 12. Microsoft. [En línea] 2017, 10 de 10 de 12. [Citado el: 08 de 09 de 2018.] https://docs.microsoft.com/es-es/dotnet/framework/wpf/advanced/xaml-syntax-in-detail.

AWS. Estructura de las recetas. [En línea]

—. 2017. Estructura de las recetas. *Estrucutra de las recetas.* [En línea] AWS, 20 de 15 de 2017. [Citado el: 12 de 07 de 2018.] https://docs.aws.amazon.com/es_es/opsworks/latest/userguide/cookbooks-101-basics-structure.html.

CHEF. 2017. ¿Por qué Chef y automatización continua? [En línea] chef, 09 de 09 de 2017. [Citado el: 09 de 09 de 2017.] https://www.chef.io/why-chef/.

—. chef. [En línea] [Citado el: 10 de 11 de 2017.] www.chef.io/why-chef.

—. chef.io. [En línea] [Citado el: 10 de 11 de 2017.] https://www.chef.io/why-chef/.

—. 2017. Chef-cliente. [En línea] 02 de 10 de 2017. [Citado el: 25 de 09 de 2018.] https://docs.chef.io/_images/chef_run.png.

—. 2016. server_components. [En línea] 26 de 07 de 2016. [Citado el: 12 de 08 de 2018.] https://docs.chef.io/_images/server_components.svg.

—. 2017. Sobre nosotros. [En línea] 09 de 09 de 2017. [Citado el: 09 de 09 de 2017.] https://www.chef.io/about/.

CHEF.IO. 2017. RECETAS. [En línea] 15 de 12 de 2017. [Citado el: 5 de 6 de 2018.] https://docs.chef.io/recipes.html.

consiste, En que. 2017. PoweData. [En línea] 2 de 07 de 2017. [Citado el: 02 de 09 de 2018.] https://www.powerdata.es/big-data.

cookbook. 2018. linke. [En línea] 19 de 11 de 2018. [Citado el: 19 de 11 de 2018.] https://blog.linkeit.com/glosario-terminos-chef-server.

DATABAGS. 2017. Revistadigital. [En línea] 10 de 08 de 2017. [Citado el: 11 de 09 de 2018.] https://revistadigital.inesem.es/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/.

DELL. 2004. DELL. [En línea] 01 de 05 de 2004. http://www.dell.com/downloads/global/power/ps2q05-20040179-Saify-OE.pdf.

dell. 2004. www.dell.com. [En línea] 01 de 05 de 2004. http://www.dell.com/downloads/global/power/ps2q05-20040179-Saify-OE.pdf.

disponibilidad, Sistemas de alta. 2018. repository.udistrila.edu.co. [En línea] 2018. [Citado el: 02 de 07 de 2018.] http://repository.udistrital.edu.co/bitstream/11349/8297/5/GuzmanMoyaPaulaAndrea2018.pdf.

Edwin Andrés Torres, William Alonso Rincón Saavedra. 2018. Repository.ucatolica.edu.co. [En línea] 2018. [Citado el: 02 de 08 de 2018.] https://repository.ucatolica.edu.co/handle/10983/18238.

ELESAPIENS. 2018. El metodo Cientifico. [En línea] LEARNIN & FUN, 06 de 02 de 2018. [Citado el: 06 de 02 de 2018.]

Erlang. 2017. ERLANG. [En línea] 10 de 11 de 2017. [Citado el: 10 de 11 de 2017.] http://www.erlang.org/about.

estructurada, Programacion. 2018. Wikipedia. [En línea] 19 de 10 de 2018. [Citado el: 20 de 10 de 2018.] https://es.wikipedia.org/wiki/Programaci%C3%B3n_estructurada.

IBM. 2016. https://www.ibm.com/developerworks/ssa/library/bd-archpatterns1/index.html. [En línea] IBM, 17 de 09 de 2016. [Citado el: 20 de 06 de 2018.] https://www.ibm.com/developerworks/ssa/library/bd-archpatterns1/index.html.

—. Ventajas de la alta disponibilidad. [En línea] IBM. [Citado el: 02 de 02 de 2018 .] https://www.ibm.com/support/knowledgecenter/es/ssw_ibm_i_73/rzarj/rzarjbenefitsha.htm.

linke. 2018. Chef-server. [En línea] 15 de 06 de 2018. [Citado el: 10 de 9 de 2018.] https://blog.linkeit.com/glosario-terminos-chef-server.

Maritza del Pilar SánchеZ, Avilio Villamiza. 2016. Modelo de Costos de Servicios de Tecnologias de inforamcion en Instituciones de Esducacion superior. *TICAL.* [En línea] 13 de 09 de 2016. [Citado el: 04 de 04 de 2018.] https://documentos.redclara.net/bitstream/10786/1085/1/Modelo%20de%20Costos%20de%20Servicios%20de%20Tecnolog%C3%ADas%20de%20Informaci%C3%B3n%20en%20Instituciones%20de%20Educaci%C3%B3n%20Superior%20%282%29.pdf.

MICROSOFT. 2018. precios y licencias. [En línea] microsoft, 15 de 02 de 2018. [Citado el: 25 de 09 de 2018.] https://www.microsoft.com/es-es/cloud-platform/windows-server-pricing.

NAVAS, ANTONIO ANTIÑOLO. Clusters. [En línea] [Citado el: 02 de 03 de 2018.] http://www.inf-cr.uclm.es/www/sbenito/AIC/Cursos%20Anteriores/Curso%202004-2005/Transparencias/Clusters.pdf.

NIST. NIST Cloud Computing Program - NCCP. [En línea] National Institute of standars and Technology U.S. [Citado el: 10 de 12 de 2017.] https://www.nist.gov/programs-projects/nist-cloud-computing-program-nccp.

OPENSTACK. Software de código abierto para la creación de nubes privadas y públicas. [En línea] [Citado el: 12 de 12 de 2017.] https://www.openstack.org/home-2?utm_expid=.OxTNoTdOSrqf1ItR3pCEsw.1&utm_referrer=https%3A%2F%2Fwww.google.com.co%2F.

PUPPET. 9 maneras en que la automatización de TI lo hace más exitoso. [En línea] [Citado el: 16 de 11 de 2017.] https://puppet.com/resources/whitepaper/9-ways-it-automation-makes-you-more-successful.

Reíta Reyes, Jorge Eduardo y Salinas Hernández, Héctor Javier. 2016. repository.udistrital.edu.co. [En línea] 2016 de 08 de 2016. [Citado el: 7 de 7 de 2018.] http://repository.udistrital.edu.co/bitstream/11349/4018/1/Big-data-FINAL-SI-1-1%20%281%29.pdf.

RUBY. Acerca de Ruby. [En línea] Ruby y jekyll. [Citado el: 07 de 10 de 2017.] https://www.ruby-lang.org/es/.

Ruby. ruby-lang.org. [En línea] Ruby. [Citado el: 15 de 11 de 2017.] https://www.ruby-lang.org/es/about/.

Scrip. 2017. webdesarrolladores. [En línea] 01 de 11 de 2017. [Citado el: 25 de 07 de 2018.] http://websarrolladores.com/2018/09/10/script/.

TICXAR, SUNQU Y. Conceptos y ecosistema big data. [En línea] [Citado el: 11 de 09 de 2017.] http://www.zemsania.com/recursos-zemsania/whitepapers/DTS/Ecosistema_Big_Data.pdf.

ANEXOS

ANEXO A
Manual de configuración e instalación chef.

Manual de instalación y configuración

CHEF

Chef software

Tabla de contenido

Tabla de imágenes

Introducción

Este manual va dirigido para todas aquellas personas que deseen implementar el uso de una herramienta de autogestión y autoconfiguración para diferentes áreas de tecnología, especializado en servicios de Big Data y alta disponibilidad.

En el manual se configuran servicios en alta disponibilidad para ecosistemas de Big Data así como la implementación del software chef en sus tres componentes chef-server, chef-Workstation y chef-cliente, desarrollando una buena metodología para todas las personas que deseen desplegar servicios con una herramienta de autogestión y autoconfiguración.

Conceptos Fundamentales

Características generales:

a. Sistema Operativo: Centos7.
b. Chef-server.
c. Chef-Workstation.
d. Apache Hadoop.
e. Apache spark.
f. Apache Cassandra.
g. MongoDB.

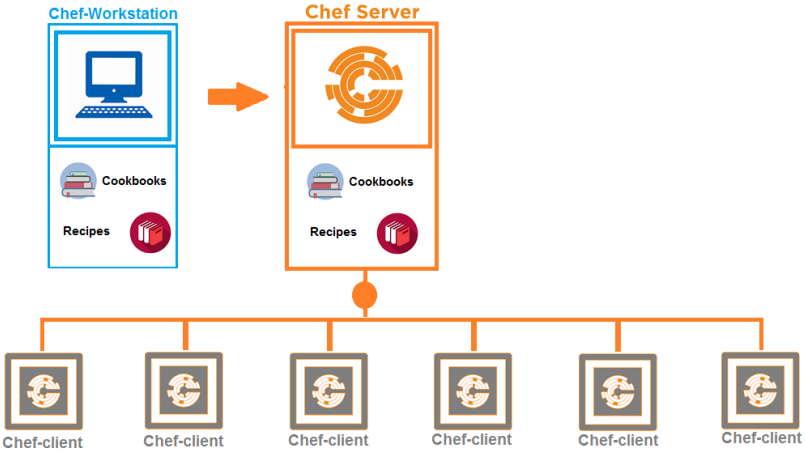
Pre-requisitos generales:

Para el correcto desarrollo de la instalación y configuración del software chef se debe tener claro los siguientes aspectos.

- Se debe tener unos conocimientos básicos en el sistema operativo Linux.
- Se debe tener unos conocimientos básicos en el lenguaje de programación Ruby.
- El equipo que sea usado para chef-server deberá contar con la memoria suficiente para soportar el número de servicios que el usuario dese instalar.

Esquema de infraestructura:

Figura 1. Chef Componentes



Instalación Chef

a. Preparación de chef-server.

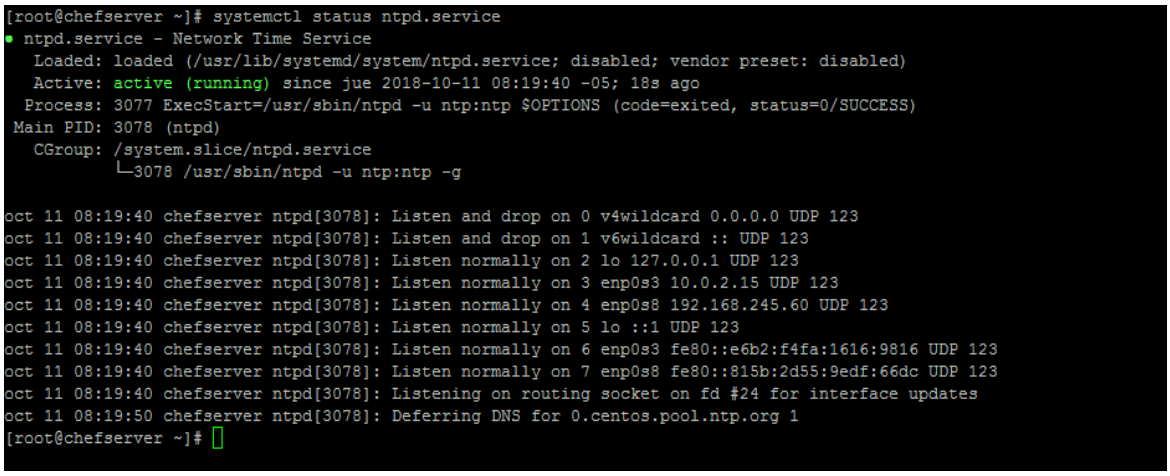
Antes de inicial con alguna descarga o configuración chef debemos verificar en el equipo que será nuestro servidor que los puestos http y https estén abiertos con el siguiente comando.

```
firewall-cmd --list-ports
firewall-cmd --permanent --add-port=80/tcp
firewall-cmd --permanent --add-port=443/tcp
firewall-cmd --reload
firewall-cmd --list-ports
```

Luego detenemos apache web server para que no tengamos problemas con algunos componentes de instalación de chef-server como el servidor web Nginx luego ejecutamos el daemon NTP para verificar que este corriendo sin problemas.

```
systemctl disable httpd
systemctl stop httpd
systemctl status ntpd.service
```

Figura 2. Systemctl status ntpd.service.



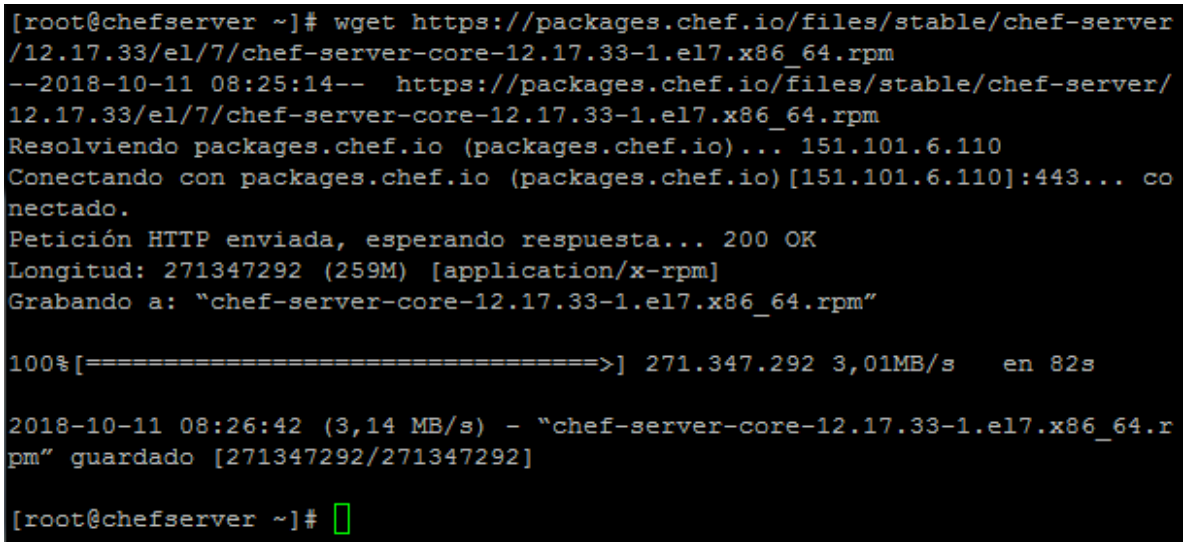
Fuente: El Autor.

b. Instalación de chef-server.

Ya todo está listo para iniciar la instalación de chef, descargaremos el paquete que chef server de su página principal www.chef-io.com y una vez descarga e instalada procederemos a reconfigurar el paquete instalado esto tomara una cantidad considerada de tiempo.

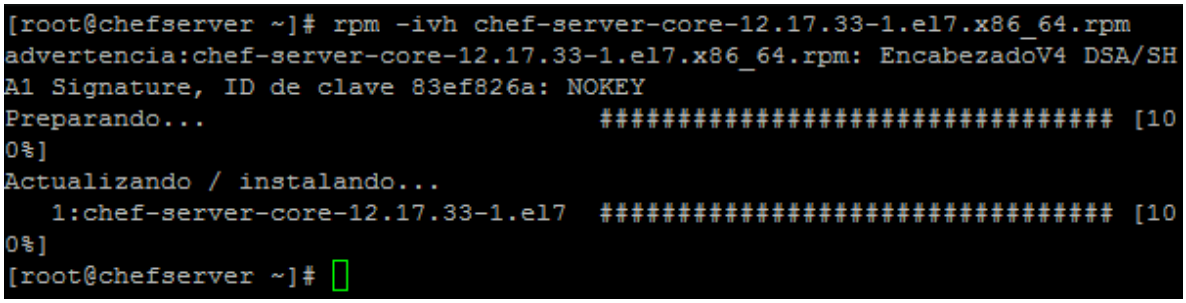
wget https://packages.chef.io/files/stable/chef-server/12.17.33/el/7/chef-server-core-12.17.33-1.el7.x86_64.rpm
rpm -ivh chef-server-core-12.17.33-1.el7.x86_64.rpm
chef-server-ctl reconfigure

Figura 3. Descarga de chef-server.



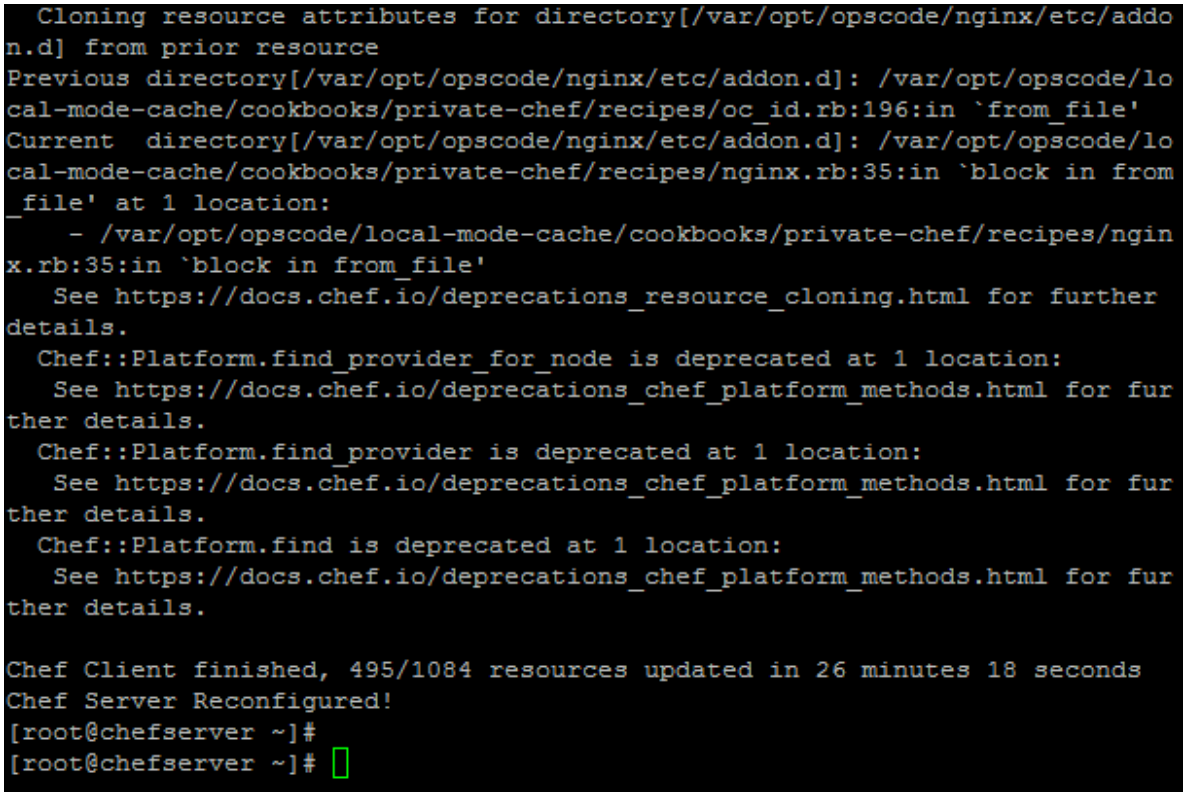
Fuente: El Autor.

Figura 4. Instalación de chef-server.



Fuente: El Autor.

Figura 5. Reconfiguración de chef.



Fuente: El Autor.

Una vez terminado de configurar el chef-server necesitaremos crear una cuenta y una organización para poder ingresar a la consola manager.

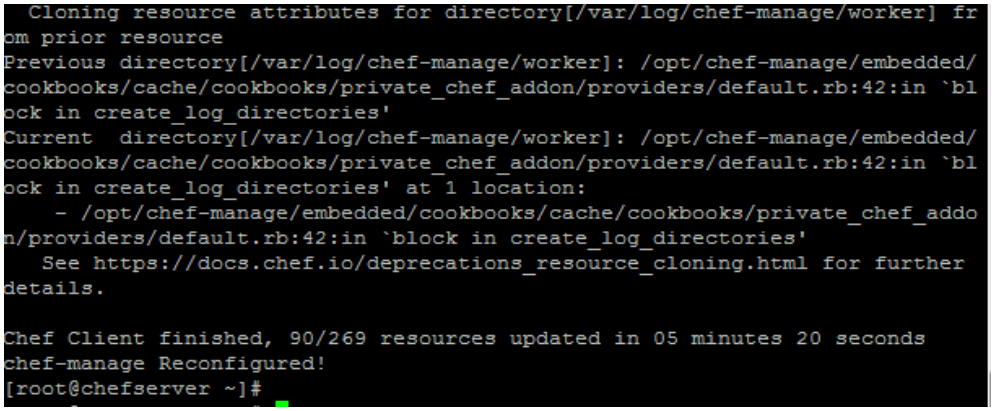
chef-server-ctl user-create ricardo fetecua puentes ricardo@chef.local ricardo -f /etc/chef/ricardo.pem

chef-server-ctl org-create tg "TG, Inc" --association_user ricardo -f /etc/chef/tg-validator.pem

Ya tenemos nuestro usuario y organización ahora procederemos a instalar el chef-manage el cual es nuestra consola principal y desde la cual se crearan y configuraran los diferentes roles de los chef-cliente que usaran las recetas y luego reconfiguramos nuevamente el chef.

chef-server-ctl install chef-manage
rpm -qa | grep chef
chef-server-ctl reconfigure
chef-manage-ctl reconfigure

Figura 6. Chef-manage reconfigured.

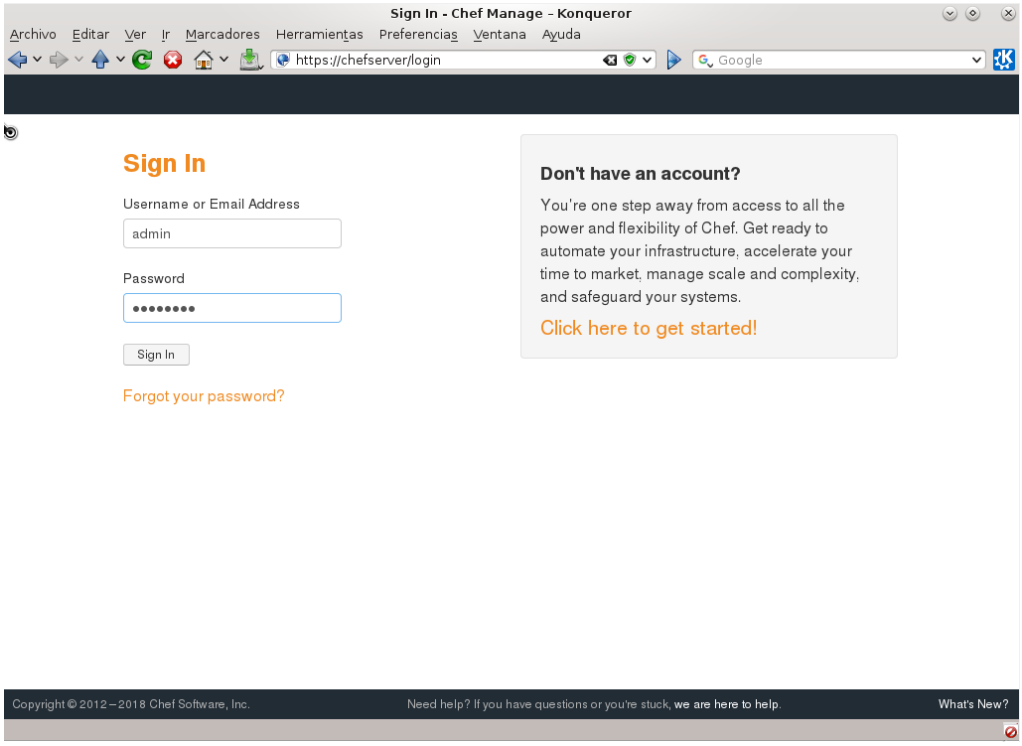


Fuente: El Autor.

Una vez termine la reconfiguración de chef podemos ingresar al browser del chef-server y verificar que todo esté bien tendrá que pedirnos el usuario y la clave que anteriormente configuramos.

https://chefserver/8080

Figura 6. Chef-Manage.



Fuente: El Autor.

Con esto solo queda ingresar usuario y contraseña y acceder al panel de configuración de chef-manager, pero por ahora instalaremos el chef-Workstation y luego ingresaremos al chef-manager para verificar que las recetas y los nodos creados puedan ser usados.

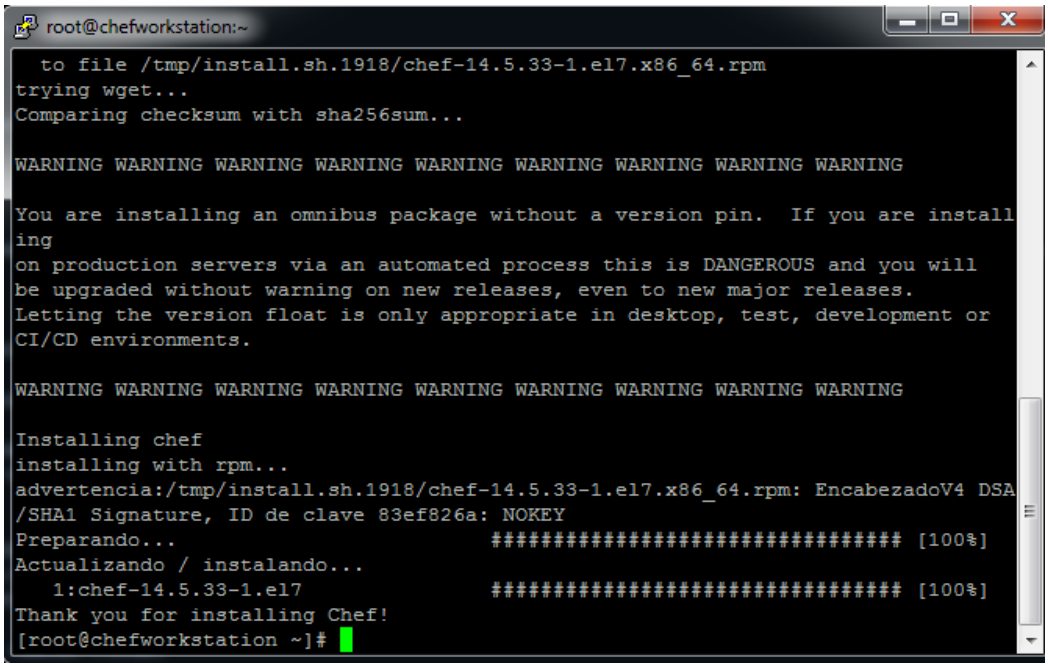
c. Instalación de Chef-Workstation.

Procedemos a descargar e instalar el paquete de instalación de chef-Workstation sobre el equipo que usaremos como estación de trabajo resaltando que tengan la misma configuración de host que el server.

Wget https://packages.chef.io/files/stable/chef/14.5.33/el/7/chef-14.5.33-1.el7.x86_64.rpm

rpm -ivh chef-14.5.33-1.el7.x86_64.rpm

Figura 20. Instalación Chef-Workstation.

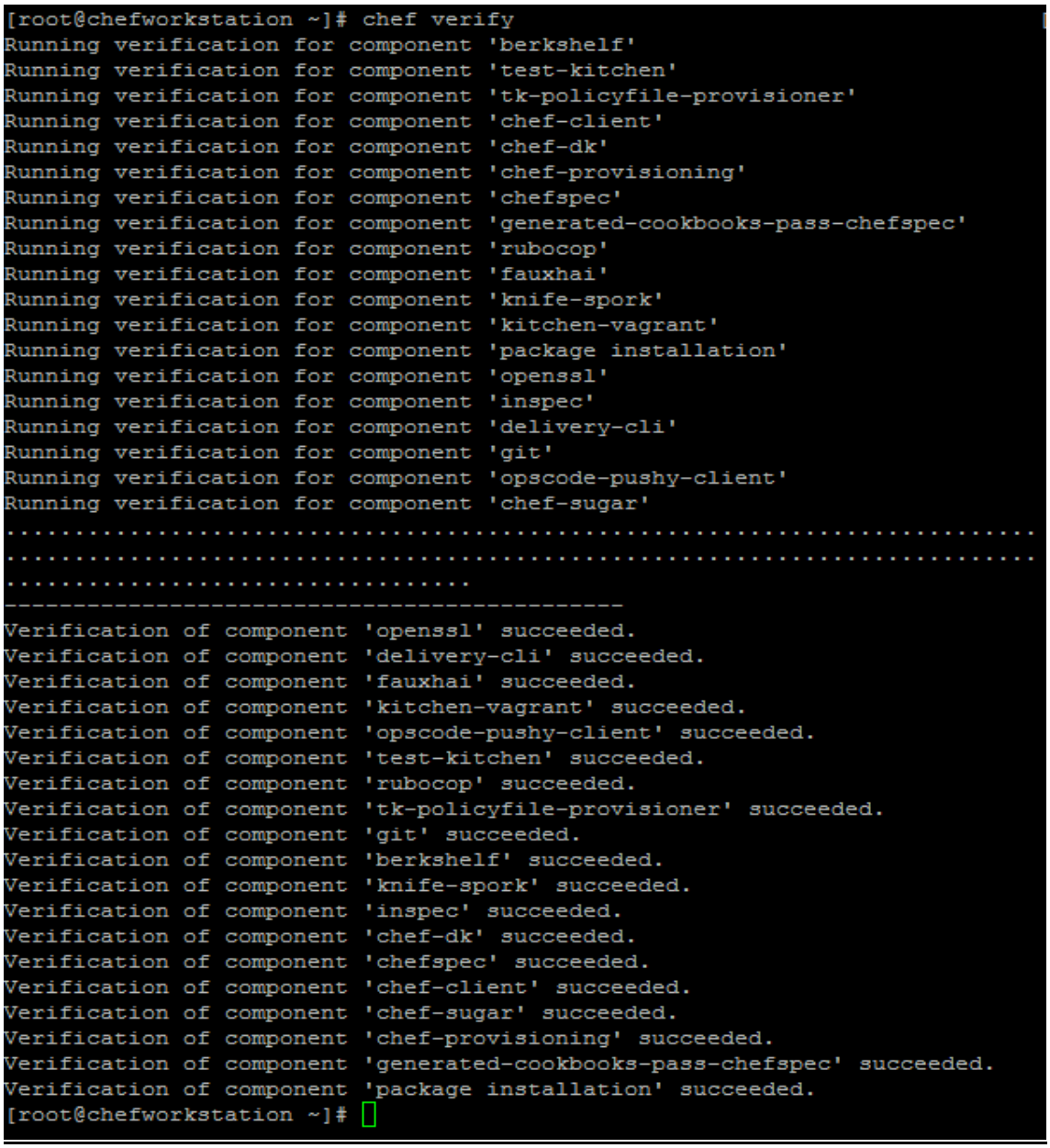


Fuente: El Autor.

Verificamos que la instalación fue exitosa con ayuda del siguiente comando.

Chef verify

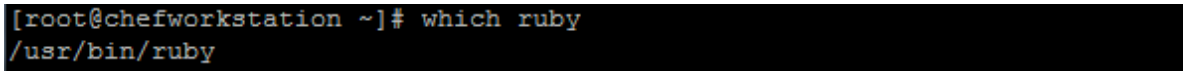
Figura 21. Chef verify.



Fuente: El Autor.

Comprobamos que la versión de Ruby este configurada adecuadamente

Figura 22. Which ruby.



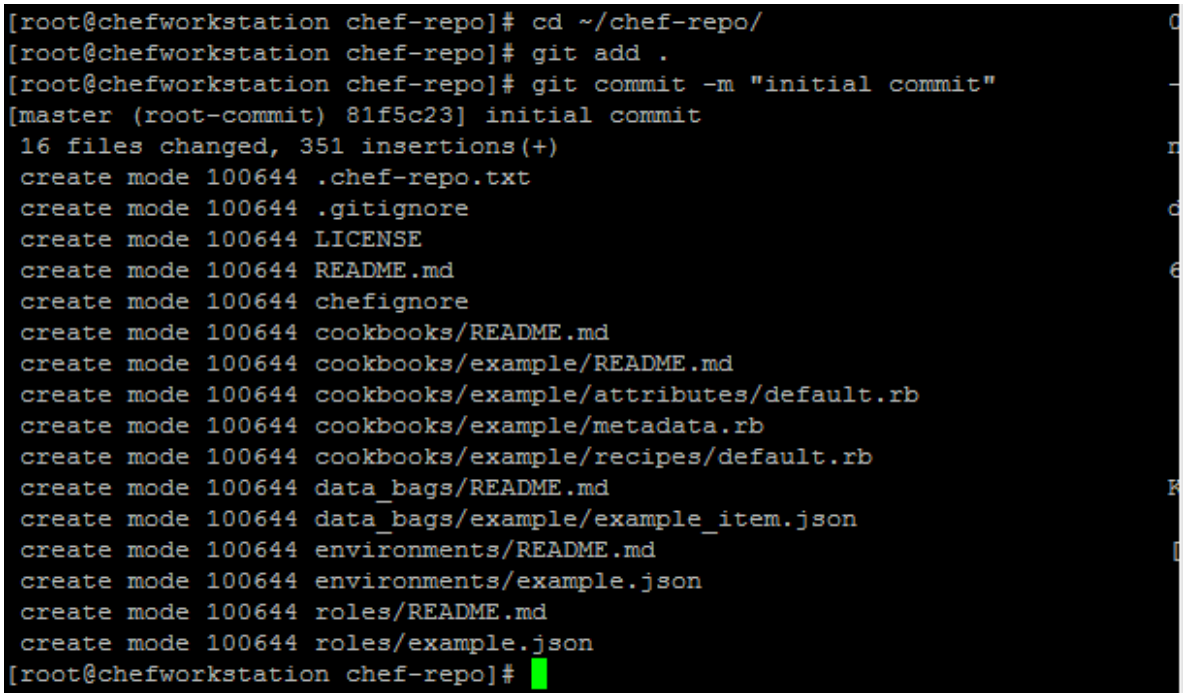
Fuente: El Autor.

d. Instalación de git

Antes de continuar con alguna otra configuración de chef instalamos una herramienta de control de versión para código abierto llamada git y la verificamos.

```
Yum –y install git
```

Figura 23. Instalación de git.



Fuente: El Autor

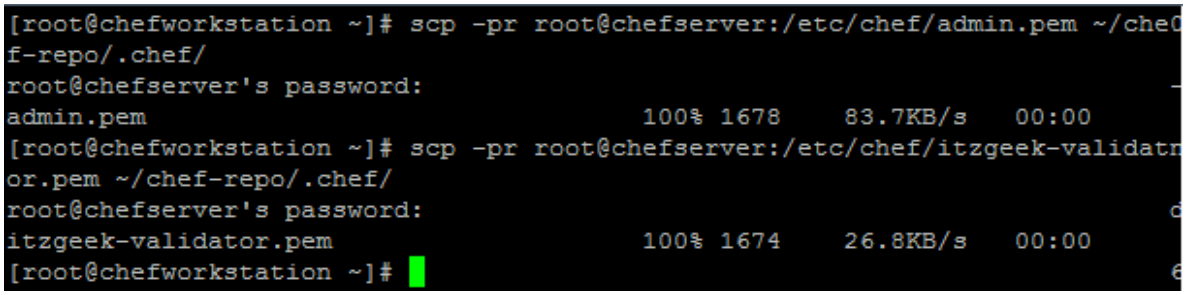
e. Copiado de llaves de seguridad.

Para que nuestra estación de trabajo se identificada por el chef-server necesitamos copiar las llaves “admin.pem” y “ORGANIZATION-validator” del servidor, con esto nuestro Workstation será reconocido por nuestro chef-server que se podrá observar en la consola de administración de la misma, esto se hace con el fin de asignar diferente estaciones de trabajo si algún usuario lo requiera solo tendrá que repetir los mismos procesos pero con un grupo diferente.

```
scp -pr root@chefserver:/etc/chef/admin.pem ~/chef-repo/.chef
```

Scp –pr root@chefserver:/etc/chef/itzgeek-validator ~/chef-repo/.chef

Figura 24.SCP chef-server.



Fuente: El Autor.

f. Knife

Para ejecutar unas buenas recetas se debe tener un instrumento el cual sea practico para ejecutar los libros de cocina es por eso que chef desarrolla un cuchillo que le permite al chef-Workstation preparar las mejores recetas y disponer de ellas de una forma organizada, para ello debemos crear knife.rb en el repositorio de chef.

node_name: nombre de usuario con permiso para autenticar en el servidor Chef.

client_key: la ubicación del archivo que contiene la clave.

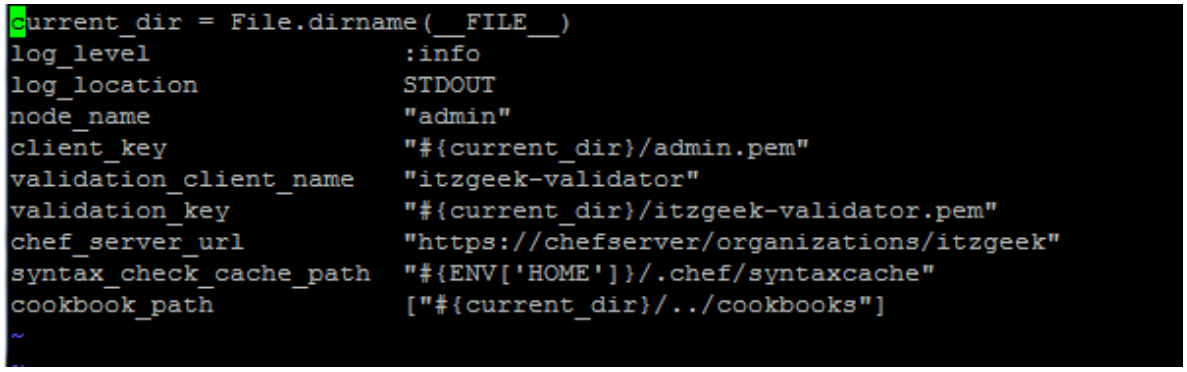
validation_client_name: el nombre de la organización seguido de -validator.

validation_key: la ubicación del archivo que contiene la clave de validación.

chef_server_url: La URL del servidor Chef. Debería comenzar con https: //, seguido de la dirección IP o FQDN del servidor Chef, el nombre de la organización al final justo después de / Organizations /.

Vi ~/chef-repo/.chef/knife.rb

Figura 25. Knife.rb.



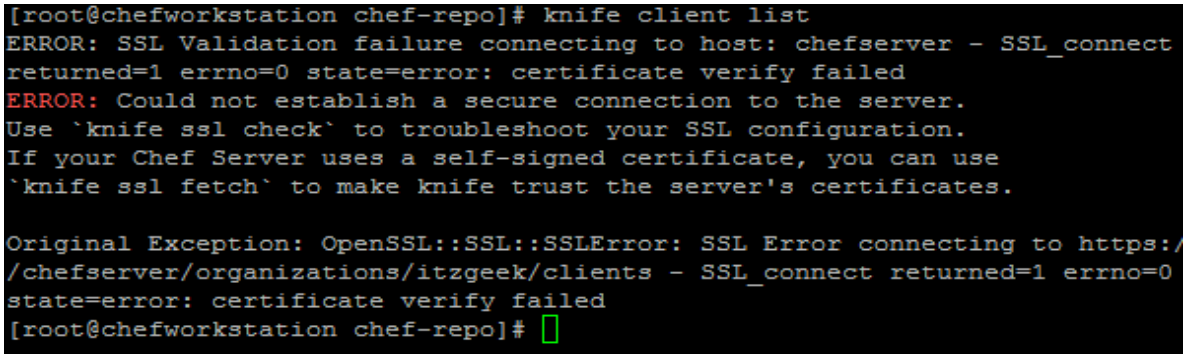
Fuente: El Autor.

g. Testing Knife.

Posterior mente a la configuración de las variables de entorno del Knife.rb probamos que todo se encuentre perfecto desde la ruta /chef-repo/, si es la primera vez que lo ejecutamos nos aparece un erros de certificados.

cd ~/chef-repo/
knife client list

Figura 26. Certificados SSL



Fuente: El Auto.

Este error se presentara cada vez que copiamos las llaves de la estación del trabajo en el servidor usando el comando

```
Knife ssl fetch
```

h. Agregar nodos.

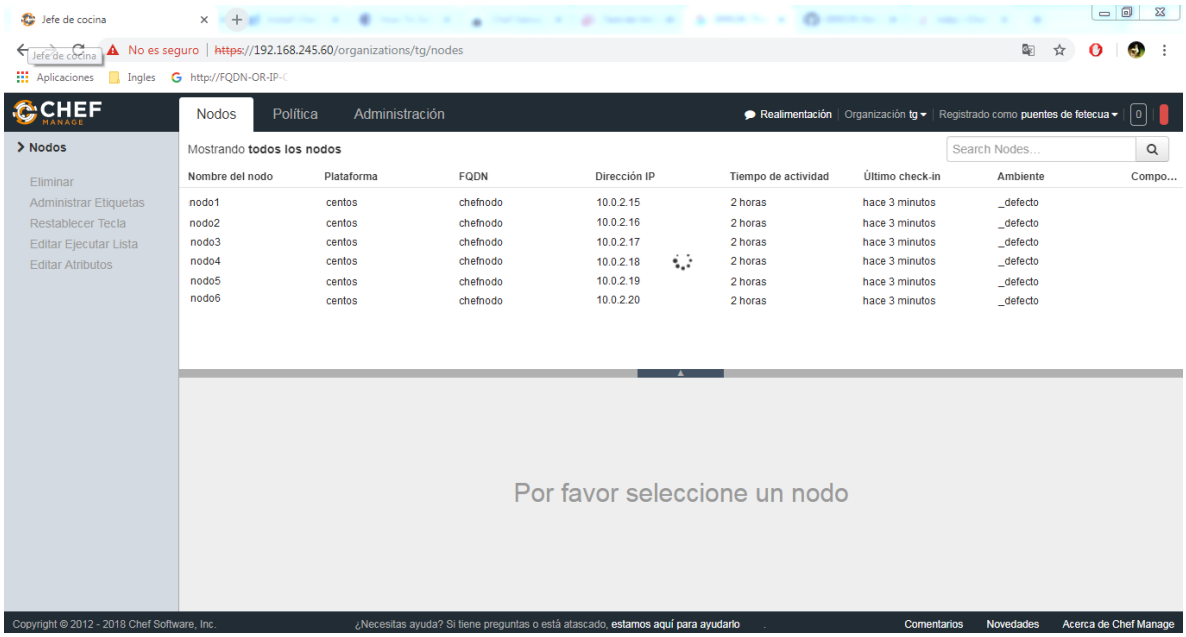
Antes de agregar nodos desde el Workstation debemos tener claro el número de clientes que ejecutaran las recetas así como la previa configuración del archivo host en este caso configuramos 6 nodos tener presente lo siguiente.

-x: el nombre de usuario ssh.
-P: La contraseña ssh.

```
knife bootstrap chefnodo -x root -P root -N nodo1 --sudo
knife bootstrap chefnodo -x root -P root -N nodo2 --sudo
knife bootstrap chefnodo -x root -P root -N nodo3 --sudo
knife bootstrap chefnodo -x root -P root -N nodo4 --sudo
knife bootstrap chefnodo -x root -P root -N nodo5 --sudo
knife bootstrap chefnodo -x root -P root -N nodo6 --sudo
```

Verificamos en nuestra consola manage con la siguiente ruta para verificar que lso nodos fueron creados “https://192.168.245.60/organizations/tg/nodes

Figura 27. Registro de nodos en consola.



Fuente: El Autor.

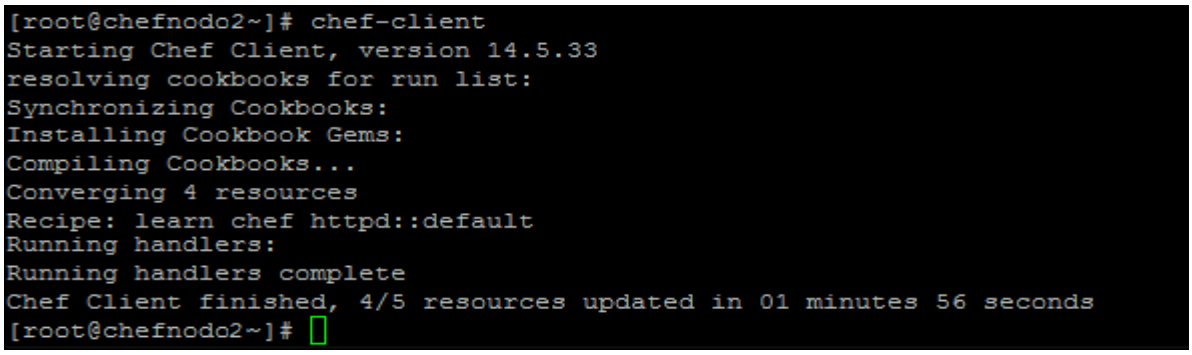
Ingresamos en uno de los nodos que se crearon en el chef-manage y ejecutamos chef-client esto nos permite ejecutar las recetas que se nos sean programadas por la estación de trabajo en primera instancia no tenemos ninguna receta agreda entonces no mostrara ningún libro de cocina o receta asociada al nodo.

Figura 28. Chef-client nodo1.

```
[root@chefnodo1~]# chef-client
Starting Chef Client, version 14.5.33
resolving cookbooks for run list:
Synchronizing Cookbooks:
Installing Cookbook Gems:
Compiling Cookbooks...
Converging 4 resources
Recipe: learn chef httpd::default
Running handlers:
Running handlers complete
Chef Client finished, 4/5 resources updated in 01 minutes 56 seconds
[root@chefnodo1~]#
```

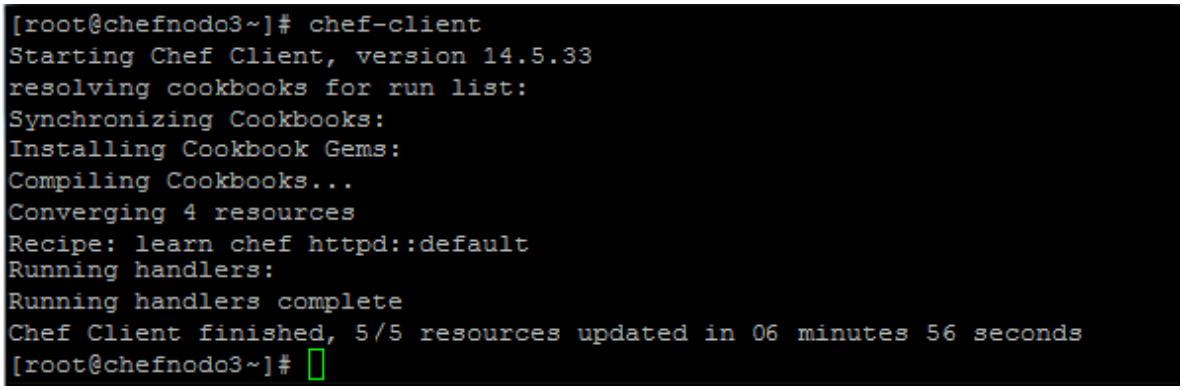
Fuente: El Autor.

Figura 29. Chef-client nodo2.



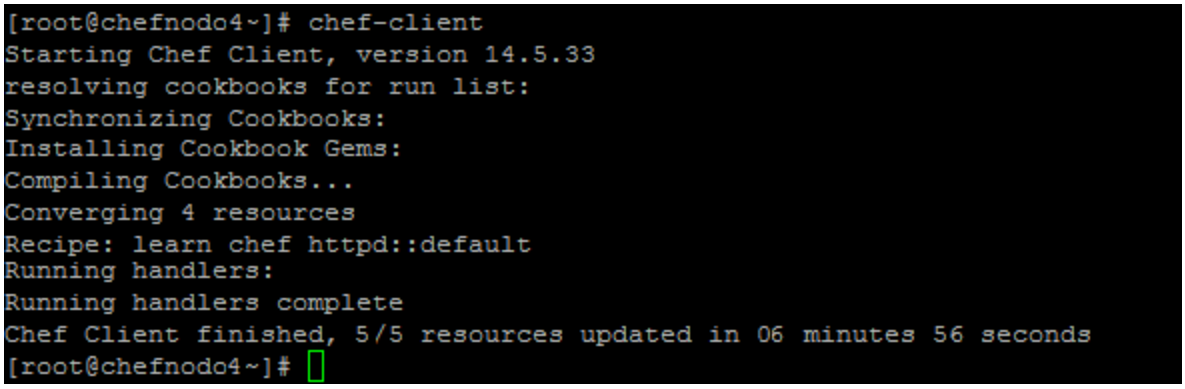
Fuente: El Autor.

Figura 30. Chef-client nodo3.



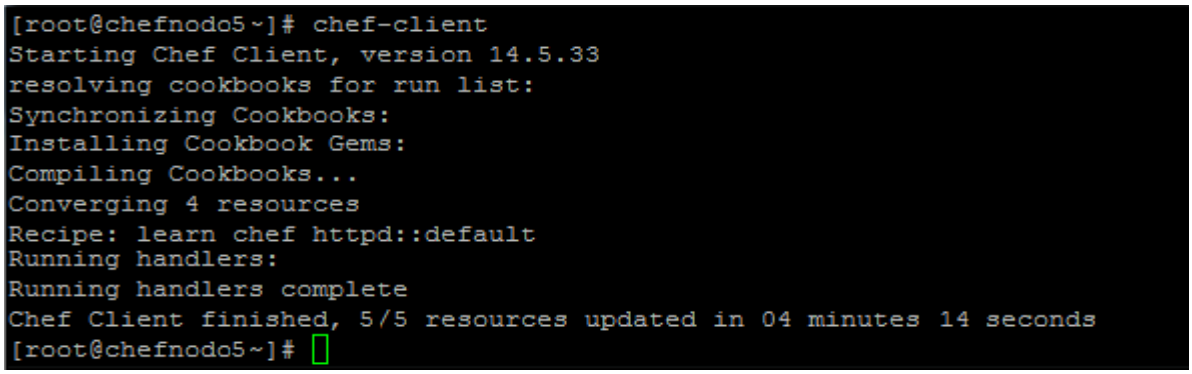
Fuente: El Autor.

Figura 31. Chef-client nodo4.



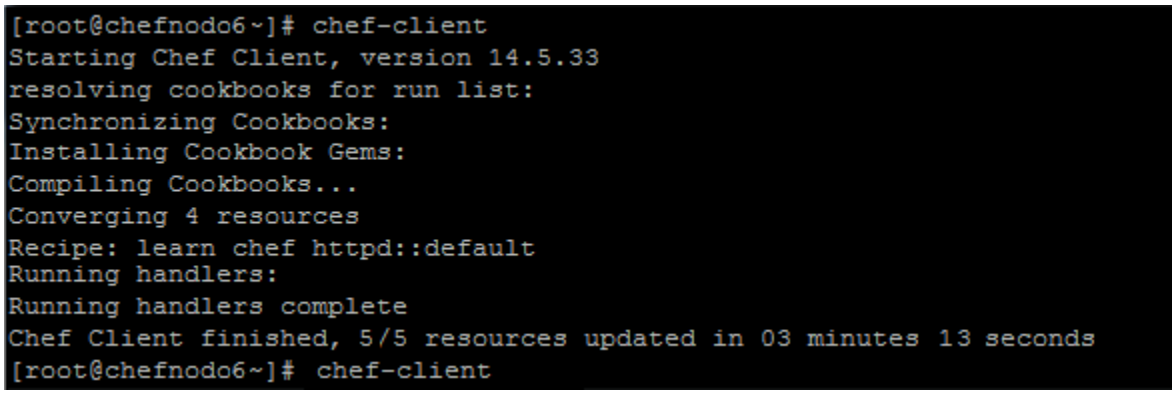
Fuente: El Autor.

Figura 32. Chef-client nodo5.



Fuente: El Autor.

Figura 33. Chef-client nodo6.



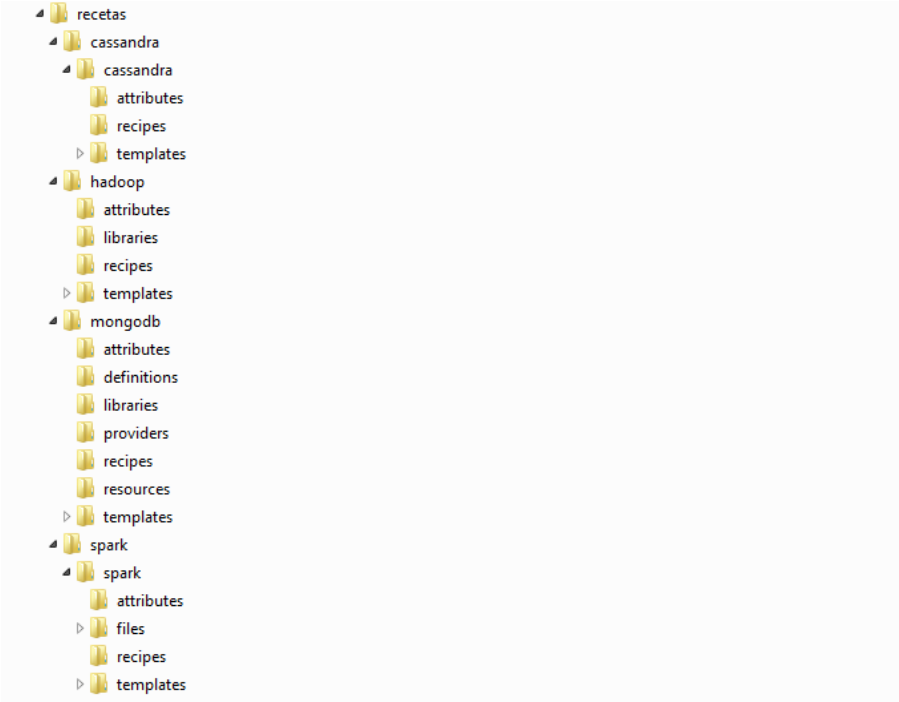
Fuente: El Autor.

i. Libro de recetas.

Ahora bien, necesitamos desplegar 4 servicios sobre la herramienta de autogestión y autoconfiguración este paso lo hacemos desde nuestra estación de trabajo quien es la encargada de configurar los roles y nodos que necesitamos para esto.

Para nuestro caso tenemos nuestros libros de recetas en nuestro equipo local es una buena práctica y solo la cargaremos usando el knife.

Figura 34. Recetas.

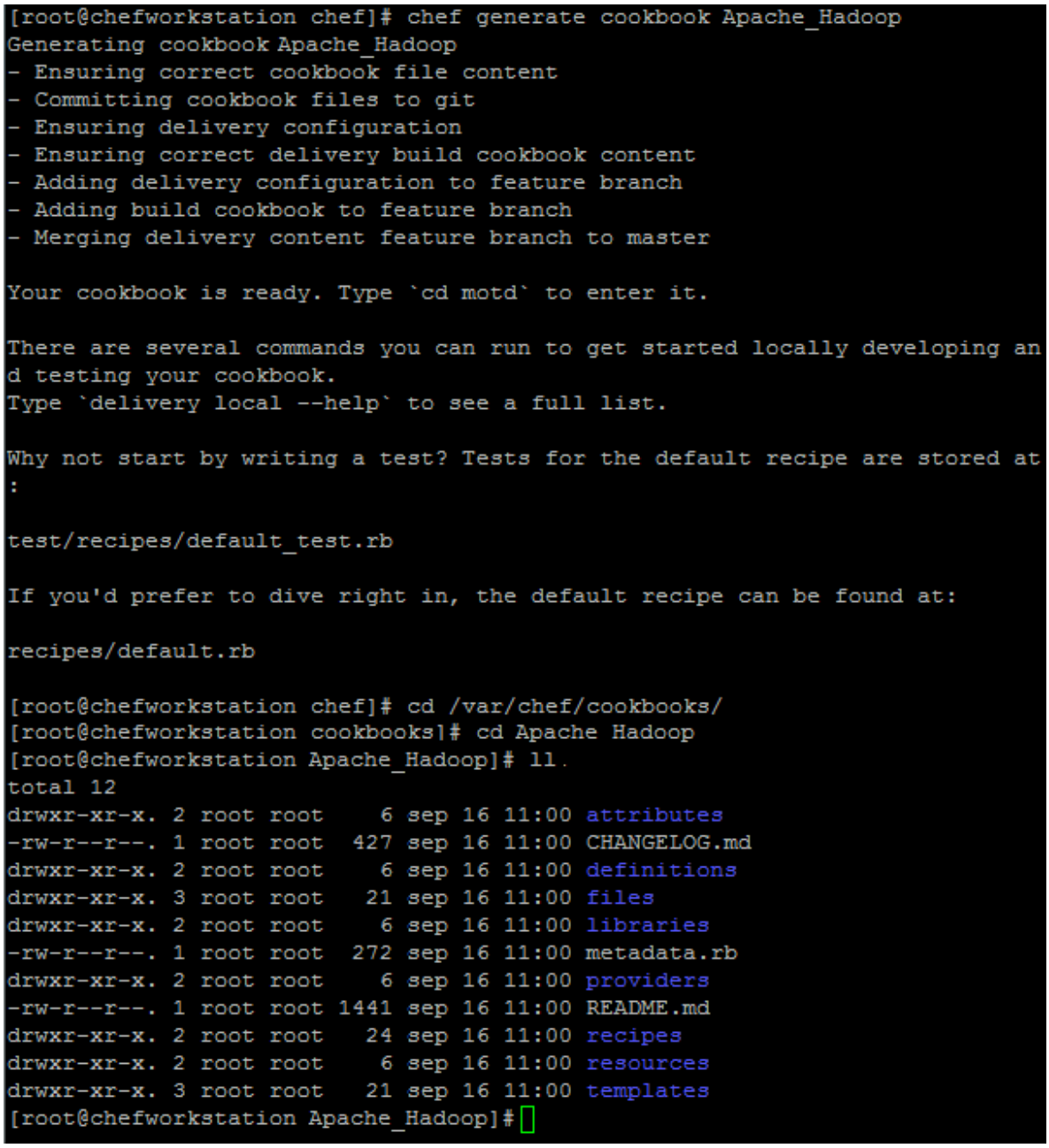


Fuente: El Autor.

Desde nuestra estación de trabajo generamos un libro de cocina con el nombre de cada servicio de se desean poner en marcha.

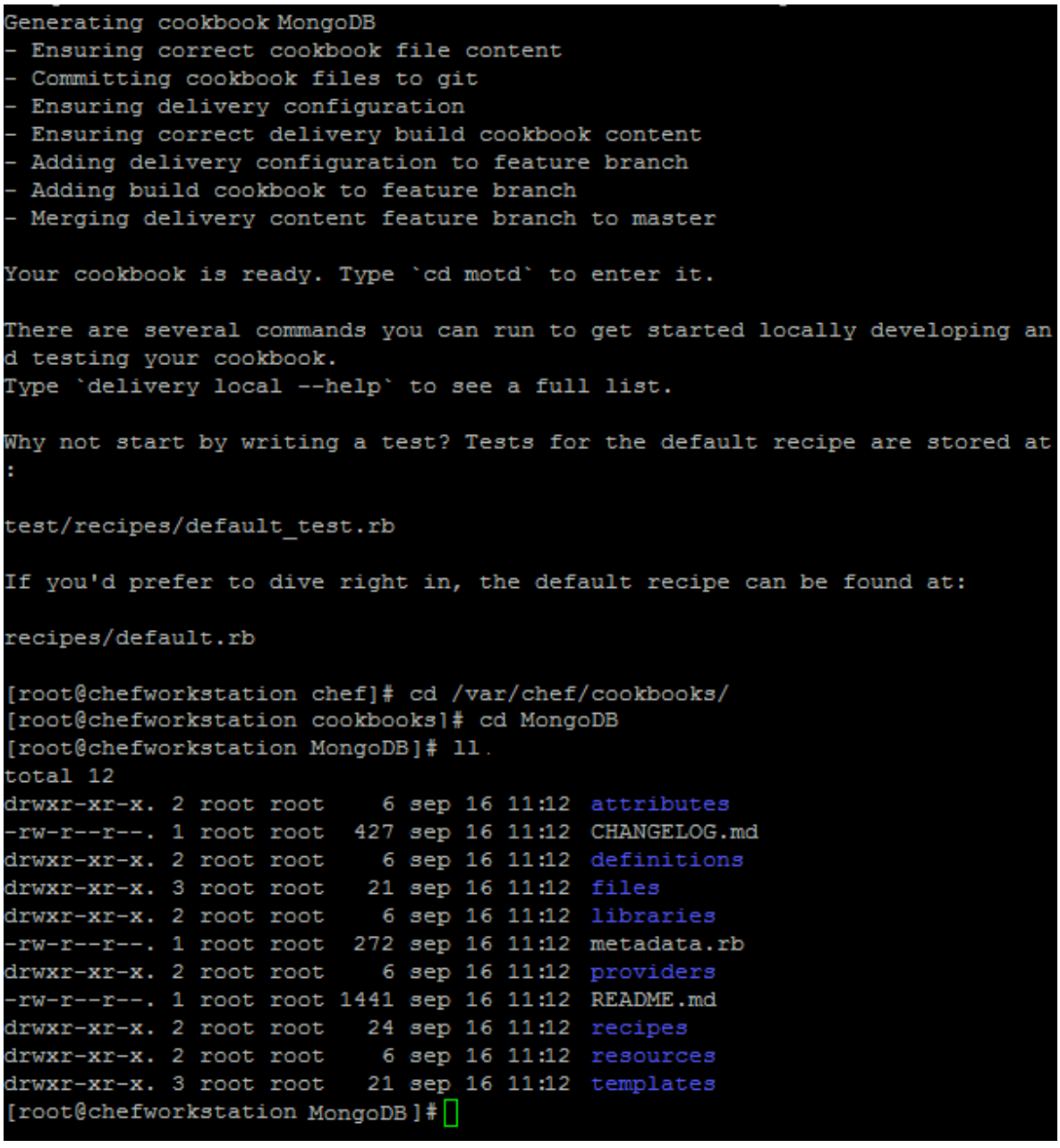
```
chef generate cookbook Apache_Hadoop
chef generate cookbook MongoDB
chef generate cookbook cassandre_apc
chef generate cookbook Spark
```

Figura 22. Chef generate cookbook Apache_Hadoop.



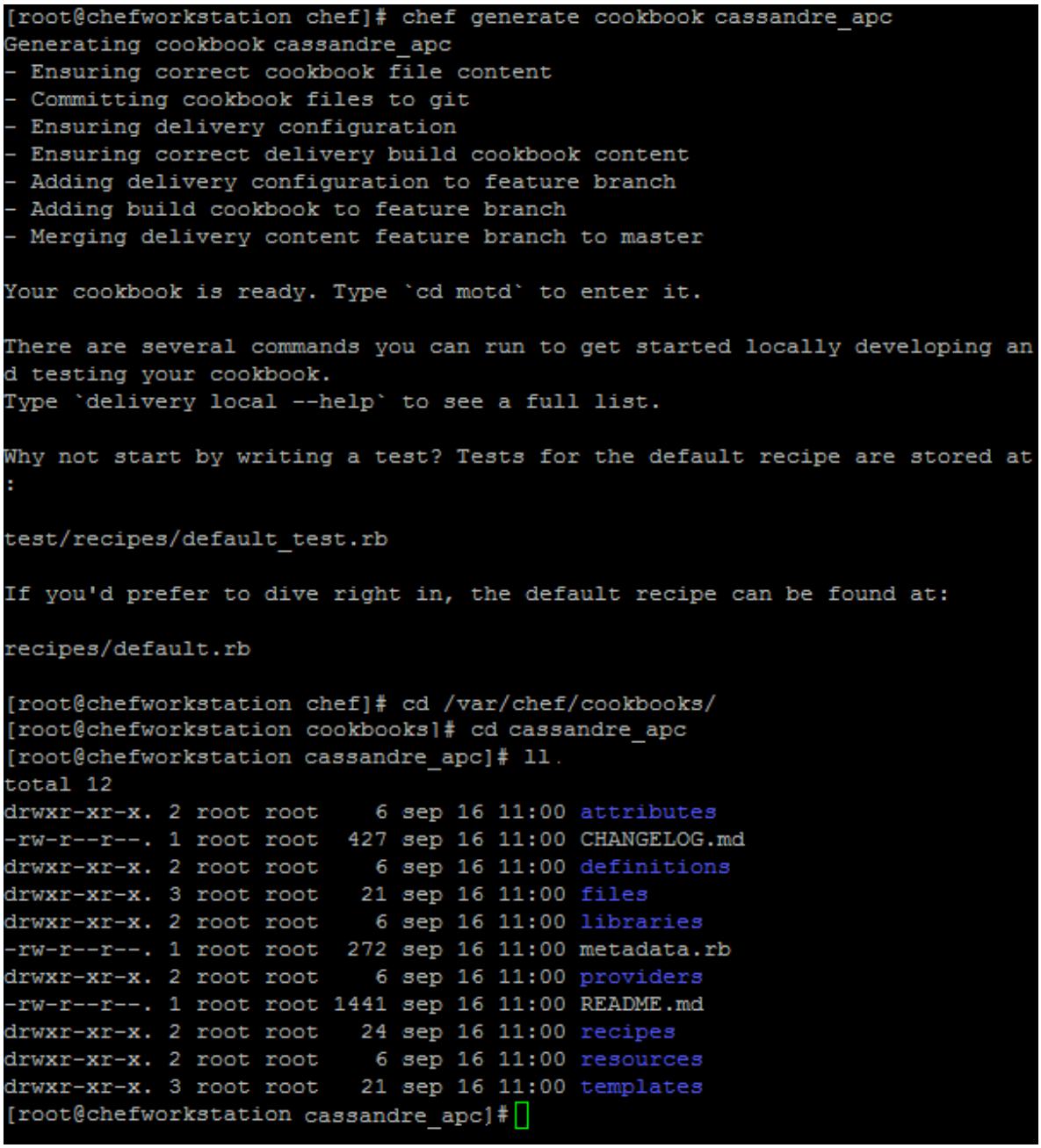
Fuente: El Autor.

Figura 23. Chef generate cookbook MongoDB.



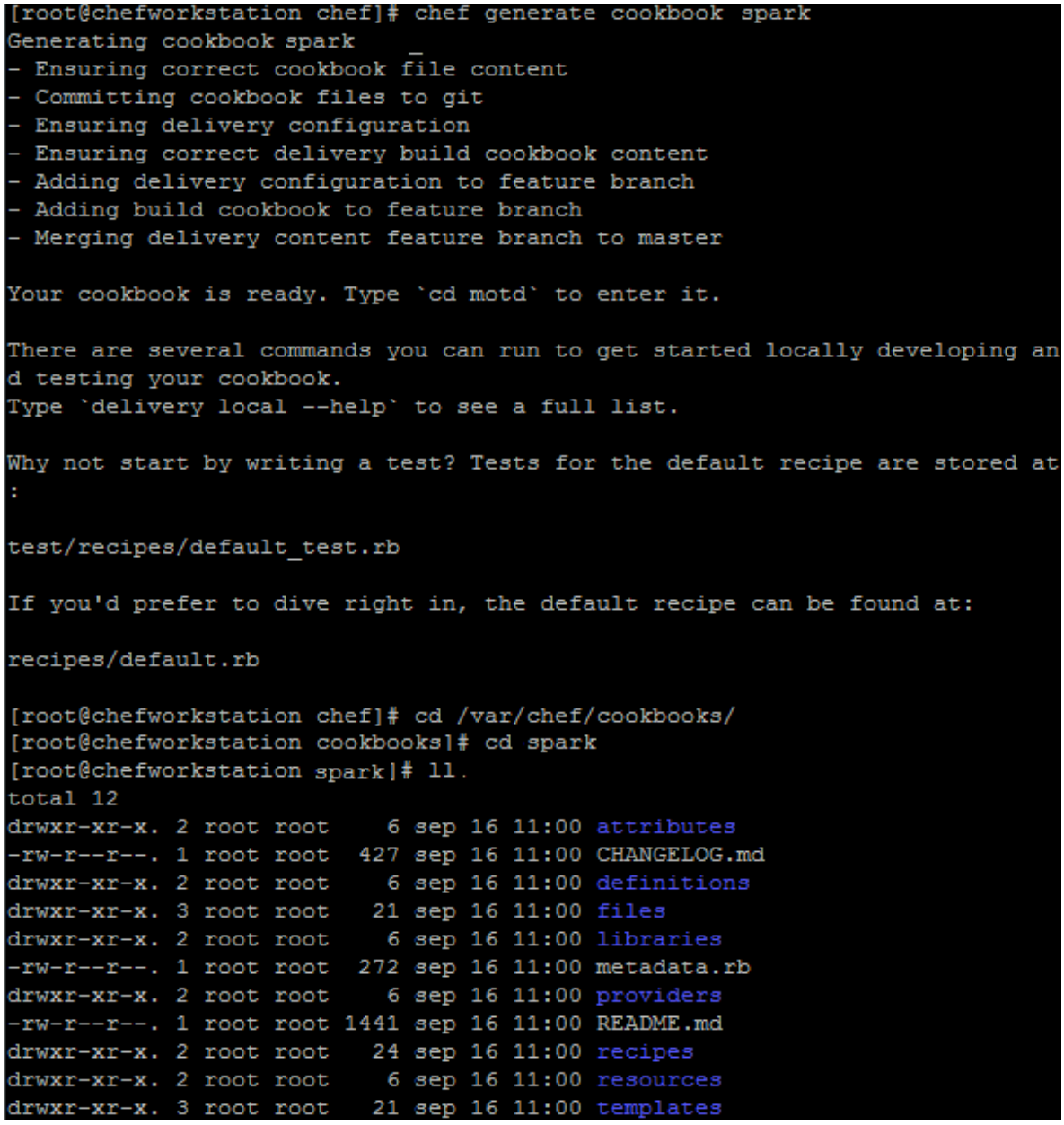
Fuente: El Autor.

Figura 35. Chef generate cookbook cassandre_apc.



Fuente: El Autor.

Figura 36. Chef generate cookbook spark

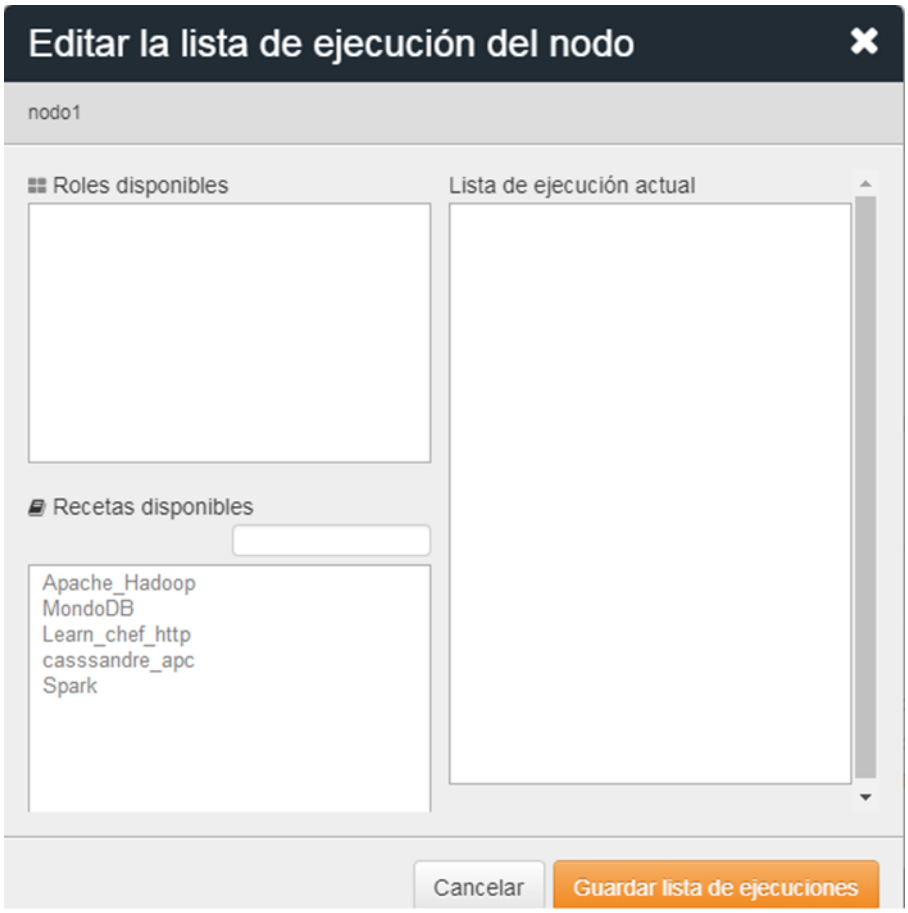


Fuente: El Autor.

Una vez cargados los servicios verificamos en la consola de administración que estén cargados los libros de cocina a un no se le ha agregado ninguna receta.

Nos debe aparecen recetas disponibles ingresando en cada nodo a un no están disponibles para un cliente.

Figura 37. Recetas Disponibles



Fuente: El Autor.

j. Cargar recetas.

Una vez tengamos creadas nuestras procedemos a cargar los archivos “default.rb” “récipes” que son los que tienen la configuración de los servicios, este paso se

hace para los 4 servicios desde la estación de trabajo y deben estar identificados con estos mismos nombres o si no el chef-cliente no podrá ejecutar los archivos.

```
$source = 'c:\cfn\downloads\Apache_Hadoop\default.rb'
$dest = 'c:\chef-repo\cookbooks\Apache_Hadoop\recipes'
Copy-Item -Path $source -Destination $dest
```

```
$source = 'c:\cfn\downloads\MongoDB\default.rb'
$dest = 'c:\chef-repo\cookbooks\MongoDB\recipes'
Copy-Item -Path $source -Destination $dest
```

```
$source = 'c:\cfn\downloads\cassandre_apc\default.rb'
$dest = 'c:\chef-repo\cookbooks\cassandre_apc\recipes'
Copy-Item -Path $source -Destination $dest
```

```
$source = 'c:\cfn\downloads\Spark\default.rb'
$dest = 'c:\chef-repo\cookbooks\Spark\recipes'
Copy-Item -Path $source -Destination $dest
```

A continuación encontramos los atributos de los cuatros servicios aclarando que se configuraron en alta disponibilidad en el que los nodos replican su información, el Scrip se puede adaptar al entorno de trabajo de cada área de infraestructura.

Figura 38. Atributos Cassandra default.rb.

```
# Creamos un usurio y un grupo para que no afecte el root del sistema operativo

default[:cassandra][:user]              = 'cassandra'
default[:cassandra][:group]             = 'nogroup'

# configuramos ip del servidor y nodos

default[:cassandra][:listen_addr]       = "cassandra1;cassandra2;cassandra3 "
default[:cassandra][:seeds]             = ["192.168.245.11"]
default[:cassandra][:rpc_addr]          = "cassandra1"
default[:cassandra][:rpc_port]          = 9160
default[:cassandra][:storage_port]      = 7000

# Instalacion
# Configuramos la version que queremos y la url de descarga

default[:cassandra][:version]           = "3.11.3"
default[:cassandra][:release_url]       = ":apache_mirror:/cassandra/:version:/apache-cassandra-:version:-bin.tar.gz"

# particiones

default[:cassandra][:max_hint_window_in_ms]        = 3600000
default[:cassandra][:hinted_handoff_delay_ms]      = 50
default[:cassandra][:auto_bootstrap]    = 'false'
default[:cassandra][:authority]         = "org.apache.cassandra.auth.AllowAllAuthority"
default[:cassandra][:partitioner]       = "org.apache.cassandra.dht.RandomPartitioner"       # "org.apache.cassandra.d
default[:cassandra][:dynamic_snitch]    = 'true'
default[:cassandra][:initial_token]     = ""
default[:cassandra][:hinted_handoff_enabled]       = 'true'

# Ahora configuramos el tamaño de almacenamiento de los componentes es importante mirar las especificaciones
# de los fabricantes

default[:cassandra][:java_heap_size_min]           = "100M"
default[:cassandra][:java_heap_size_max]           = "2000M"
default[:cassandra][:java_heap_size_eden]          = "2500M"
default[:cassandra][:disk_access_mode]             = "auto"
default[:cassandra][:concurrent_reads]             = 8
default[:cassandra][:concurrent_writes]            = 32
default[:cassandra][:memtable_flush_writers]       = 1
default[:cassandra][:flush_largest_memtables_at]   = 0.75
default[:cassandra][:reduce_cache_sizes_at]        = 0.85
default[:cassandra][:rpc_timeout_in_ms]            = 10000
default[:cassandra][:rpc_keepalive]                = "false"
default[:cassandra][:phi_convict_threshold]        = 8
default[:cassandra][:request_scheduler]            = 'org.apache.cassandra.scheduler.NoScheduler'
default[:cassandra][:throttle_limit]               = 80
default[:cassandra][:request_scheduler_id]         = 'keyspace'

default[:tuning][:ulimit]['cassandra'] = { :nofile => { :both => 32768 }, :nproc => { :both => 50000 } }
```

Fuente: El Autor.

Figura 39. Atributos Hadoop default.rb.

```
#configuracion inicial de cluster hadoop
default['hadoop']['replicacion'] = 'hdp'
# valores por defecto para los nodos

default['hadoop']['replicacion_version'] =
  if node['hadoop'].key?('replicacion_version')
    node['hadoop']['replicacion_version']
  elsif node['hadoop']['replicacion'] == 'hdp'
    '2.4.3.0'
  elsif node['hadoop']['replicacion'] == 'cdh'
    '5.7.3'
  elsif node['hadoop']['replicacion'] == 'bigtop'
    '1.0.0'
  elsif node['hadoop']['replicacion'] == 'iop'
    '4.1.0.0'
  end

default['hadoop']['force_format'] = false
# archivos de configuracion de hadoop

default['hadoop']['conf_dir'] = 'conf.chef'
default['flume']['conf_dir'] = node['hadoop']['conf_dir']
default['hadoop_kms']['conf_dir'] = node['hadoop']['conf_dir']
default['hbase']['conf_dir'] = node['hadoop']['conf_dir']
default['hive']['conf_dir'] = node['hadoop']['conf_dir']
default['hive2']['conf_dir'] = node['hadoop']['conf_dir']
default['oozie']['conf_dir'] = node['hadoop']['conf_dir']
default['spark']['conf_dir'] = node['hadoop']['conf_dir']
default['spark2']['conf_dir'] = node['hadoop']['conf_dir']
default['storm']['conf_dir'] = node['hadoop']['conf_dir']
default['tez']['conf_dir'] = node['hadoop']['conf_dir']
default['zookeeper']['conf_dir'] = node['hadoop']['conf_dir']

# Configurar ip local del servidor
default['hadoop']['sysctl']['net.ipv4.ip_local_reserved_ports'] = []
# configuracion core-site.xml
default['hadoop']['core_site']['fs.defaultFS'] = "hdfs://#{node['fqdn']}"
# configuracion yarn-site.xml
default['hadoop']['yarn_site']['yarn.resourcemanager.hostname'] = node['fqdn']
default['hadoop']['yarn_site']['yarn.scheduler.increment-allocation-vcores'] = '1'
# Establecemos los entornos de hadoop
default['hadoop']['yarn_site']['yarn.application.classpath'] =
  if node['hadoop']['replicacion'] == 'hdp' && node['hadoop']['replicacion_version'].to_f >= 2.2
    '/etc/hadoop/conf,/usr/hdp/current/hadoop-client/*,/usr/hdp/current/hadoop-client/lib/*,/usr/hdp/current/ha
  else
    '$HADOOP_CONF_DIR, $HADOOP_COMMON_HOME/*, $HADOOP_COMMON_HOME/lib/*, $HADOOP_HDFS_HOME/*, $HADOOP_HDFS_HOME
  end

# definimos la replicacion para 4 nodos

if node['hadoop']['replicacion'] == 'cdh' && node['hadoop']['replicacion_version'].to_i == 4

  default['hadoop']['yarn_site']['yarn.nodemanager.aux-services'] = 'mapreduce.shuffle'
  default['hadoop']['yarn_site']['yarn.nodemanager.aux-services.mapreduce.shuffle.class'] = 'org.apache.hadoop.
else
  default['hadoop']['yarn_site']['yarn.nodemanager.aux-services'] = 'mapreduce_shuffle'
  default['hadoop']['yarn_site']['yarn.nodemanager.aux-services.mapreduce_shuffle.class'] = 'org.apache.hadoop.
end
```

Fuente: El Autor.

Figura 40. Atributos MongoDB default.rb.

```
# configuramos los nodos y los nombresdel cluster
default[:mongodb][:client_roles] = [3]
default[:mongodb][:cluster_name] = nil
default[:mongodb][:shard_name] = 'default'

# habilitamos la replicacion de datos para los nodos
default[:mongodb][:replica_arbiter_only] = true
default[:mongodb][:replica_build_indexes] = true
default[:mongodb][:replica_hidden] = true

# habilitamos 2 nodos sclavos
default[:mongodb][:replica_slave_delay] = 3
default[:mongodb][:replica_priority] = 3
default[:mongodb][:replica_tags] = {}
default[:mongodb][:replica_votes] = 3

# creamos un usuario mongodb para que no afecte el root del sistema operativo
default[:mongodb][:root_group] = 'root'
default[:mongodb][:user] = 'mongodb'
default[:mongodb][:group] = 'mongodb'

# Configuramos las rutas de instalacion
default[:mongodb][:init_dir] = '/etc/init.d'
default[:mongodb][:sysconfig_file] = '/etc/default/mongodb'
default[:mongodb][:sysconfig_file_template] = 'mongodb.sysconfig.erb'
default[:mongodb][:dbconfig_file_template] = 'mongodb.conf.erb'
default[:mongodb][:dbconfig_file] = '/etc/mongodb.conf'

# Usamos el metodo de instalacion distribuido para la replicaciones de informacion
# en los nodos 'distro' una vez configurada reiniciamos los nodos

default[:mongodb][:install_method] = 'distro'
default[:mongodb][:is_replicaset] = true
default[:mongodb][:is_shard] = true
default[:mongodb][:is_configserver] = true
default[:mongodb][:reload_action] = 'restart'

case node['platform_family']
when 'freebsd'
  default[:mongodb][:package_name] = 'mongo-10gen-server'
  default[:mongodb][:sysconfig_file] = '/etc/rc.conf.d/mongodb'
  default[:mongodb][:init_dir] = '/usr/local/etc/rc.d'
  default[:mongodb][:root_group] = 'wheel'
when 'rhel', 'fedora'

  default[:mongodb][:package_name] = 'mongodb-server'
  default[:mongodb][:sysconfig_file] = '/etc/sysconfig/mongodb'
  default[:mongodb][:user] = 'mongod'
  default[:mongodb][:group] = 'mongod'
  default[:mongodb][:init_script_template] = 'redhat-mongodb.init.erb'
  default[:mongodb][:default_init_name] = 'mongod'
  default[:mongodb][:instance_name] = 'mongod'

  default[:mongodb][:install_method] = 'mongodb-org'
  default[:mongodb][:package_name] = 'mongodb-org'
  default[:mongodb][:template_cookbook] = 'mongodb'
  default[:mongodb][:key_file_content] = nil

}
```

Fuente: El Autor

Figura 41. Atributos spark default.rb.

```
# configuramos las vercion de descar y el link de descarga las configuraciones la hacemos
# desde las recetas

default['spark']['version'] = "0.6.1"
default['spark']['download_url'] = "http://github.com/downloads/mesos/spark"
default['spark']['install_dir'] = "/opt/spark"

# como sera en alta disponibilidad formamos replicacion entre ellos.

default['spark']['mem']['master'] = "spark"
default['spark']['mem']['slave1'] = "spark"
default['spark']['mem']['slave2'] = "spark"
```

Fuente: El Autor.

Una vez cargados los atributos de cada servicio se deben subir las recetas en ocasiones se pueden manejar los pasos de configuración en los atributos y se invocados desde las recetas.

Figura 42. Receta Cassandra.

```

# receta

include_recipe "metachef"
include_recipe "volumes"
include_recipe "java" ; complain_if_not_sun_java(:cassandra)
include_recipe "thrift"

# usuario

daemon_user(:cassandra) do
  create_group  false
end

# directorio

standard_dirs('cassandra') do
  directories   [:conf_dir, :log_dir, :lib_dir, :pid_dir, :data_dirs, :commitlog_dir
  group         'root'
end
```

Fuente: El Autor.

Figura 43. Receta Hadoop.

```
1
2   include_recipe 'hadoop::repo'
3   include_recipe 'hadoop::_hadoop_checkconfig'
4   include_recipe 'hadoop::_compression_libs'
5
6   package hadoop_package('hadoop-client') do
7     action :install
8   end
9
10  package libhdfs do
11    action :install
12  end
13
14  hadoop_conf_dir = "/etc/hadoop/#{node['hadoop']['conf_dir']}"
15
16  directory hadoop_conf_dir do
17    mode '0755'
18    owner 'root'
19    group 'root'
20    action :create
21    recursive true
22  end
23
24  # configuracion de core-site.xml;hdfs-site.xml;yarn-site.xml
25  %w(capacity_scheduler core_site hadoop_policy hdfs_site mapred_site yarn_site).each do |sitefile
26    template "#{hadoop_conf_dir}/#{sitefile.tr('_', '-')}.xml" do
27      source 'generic-site.xml.erb'
28      mode '0644'
29      owner 'root'
30      group 'root'
31      action :create
32      variables options: node['hadoop'][sitefile]
33      only_if { node['hadoop'].key?(sitefile) && !node['hadoop'][sitefile].empty? }
34    end
35  end
36
37  template fair_scheduler_file.gsub('file://', '') do
38    source 'fair-scheduler.xml.erb'
39    mode '0644'
40    owner 'root'
41    group 'root'
42    action :create
43    variables node['hadoop']['fair_scheduler']
44    only_if { node['hadoop'].key?('fair_scheduler') && !node['hadoop']['fair_scheduler'].empty? }
```

Fuente: El Autor.

Figura 44. Receta MongoDB.

```
1
2   include_recipe 'mongodb::install'
3
4   allow_mongodb_instance_run = true
5   conflicting_recipes = %w(mongodb::replicaset mongodb::shard mongodb::configserver m
6   chef_major_version = Chef::VERSION.split('.').first.to_i
7 if chef_major_version < 11
8     conflicting_recipes.each do |recipe|
9       allow_mongodb_instance_run &&= false if node.recipe?(recipe)
10    end
11  else
12    conflicting_recipes.each do |recipe|
13      allow_mongodb_instance_run &&= false if node.run_context.loaded_recipe?(recipe)
14    end
15  end
16
17 if allow_mongodb_instance_run
18    mongodb_instance node['mongodb']['instance_name'] do
19      mongodb_type 'mongod'
20      bind_ip      node['mongodb']['config']['bind_ip']
21      port         node['mongodb']['config']['port']
22      logpath      node['mongodb']['config']['logpath']
23      dbpath       node['mongodb']['config']['dbpath']
24      enable_rest  node['mongodb']['config']['rest']
25      smallfiles   node['mongodb']['config']['smallfiles']
26    end
27  end
```

Fuente: El Autor

Figura 45. Receta spark.

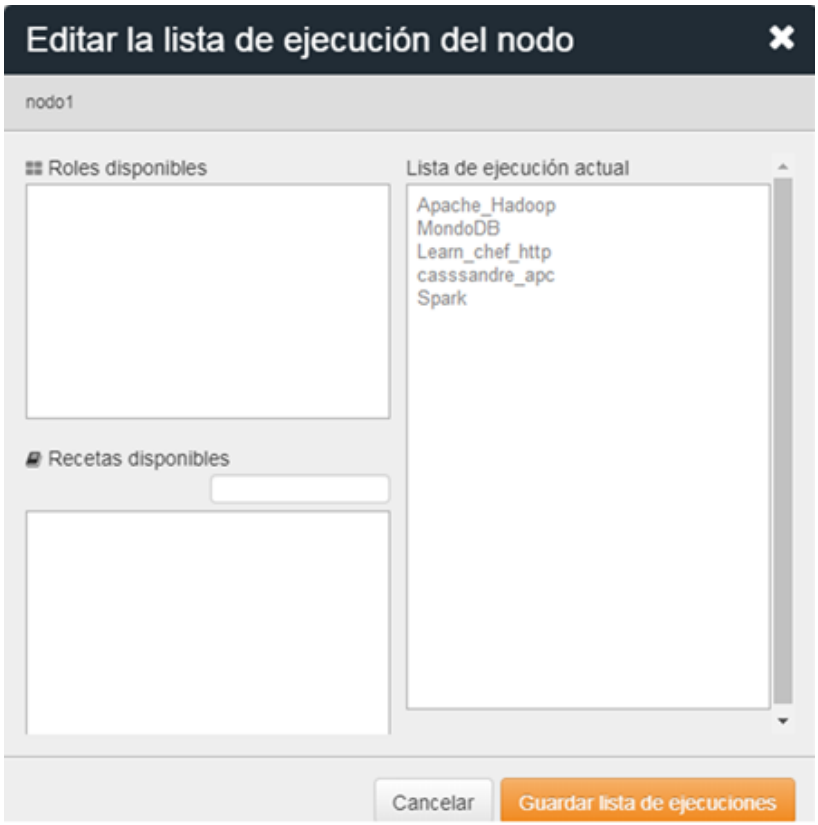
```
1
2   include_recipe "java"
3   package "scala"
4
5   #configuramos nodos y esclavos
6
7   spark_slaves = search(:node, "role:spark_slave AND chef_environment:#{node.chef_environment}")
8   spark_slaves = search(:node, "role:spark_slave AND chef_environment:#{node.chef_environment}")
9   spark_master = search(:node, "role:spark_master AND chef_environment:#{node.chef_environment}").first
10
11  # directorio de instalacion para los nodos
12  install_dir = "#{node['spark']['install_dir']}/spark-#{node['spark']['version']}"
13  mem = "#{node['spark']['mem']['slave']}"
14  if node.run_list.include?("role[spark_master]")
15    mem = "#{node['spark']['mem']['master']}"
16  end
17
18  #inicamos el usuario spark
19
20  user "spark" do
21      comment "Spark user"
22      gid "spark"
23      home "/home/spark"
24      shell "/bin/bash"
25      supports :manage_home => true
26  end
27
28  remote_directory "/home/spark/.ssh" do
29      source "ssh"
30      owner "spark"
31      group "spark"
32      files_owner "spark"
33      files_group "spark"
34      files_mode 00600
35      mode 00700
36  end
37
38  # copiamos los directorios en los nodos
39
40  directory node['spark']['install_dir'] do
41    owner "spark"
42    group "spark"
43    action :create
44    recursive true
45  end
```

Fuente: El Autor.

Para finalizar Ingresando a la consola manager de chef-server ahora si arrastramos las recetas que queremos que cada nodo ejecute una vez esto en los nodos clientes descargamos las recetas y ellas se encargara de toda la instalación este paso se repite con todos los nodos creados ejecutando en los nodos asignados el comando chef cliente

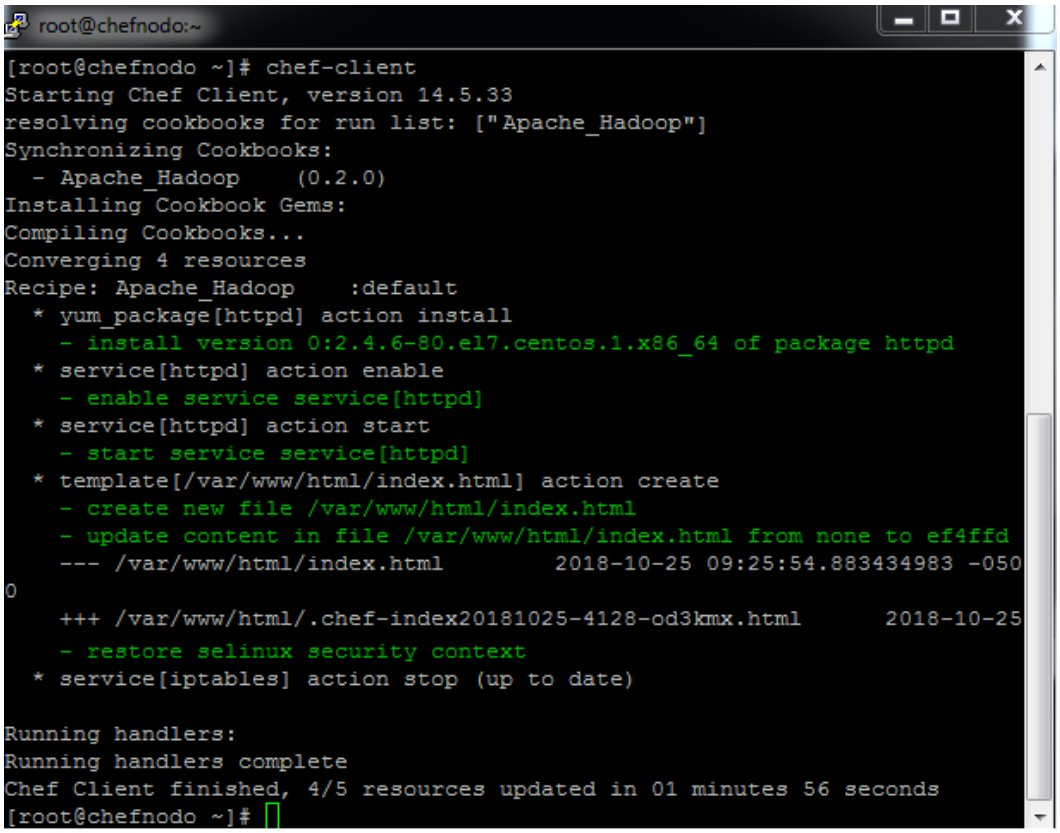
Chef-client

Figura 35. Lista de ejecución actual.



Fuente: El Autor

Figura 46 Instalación de servicios

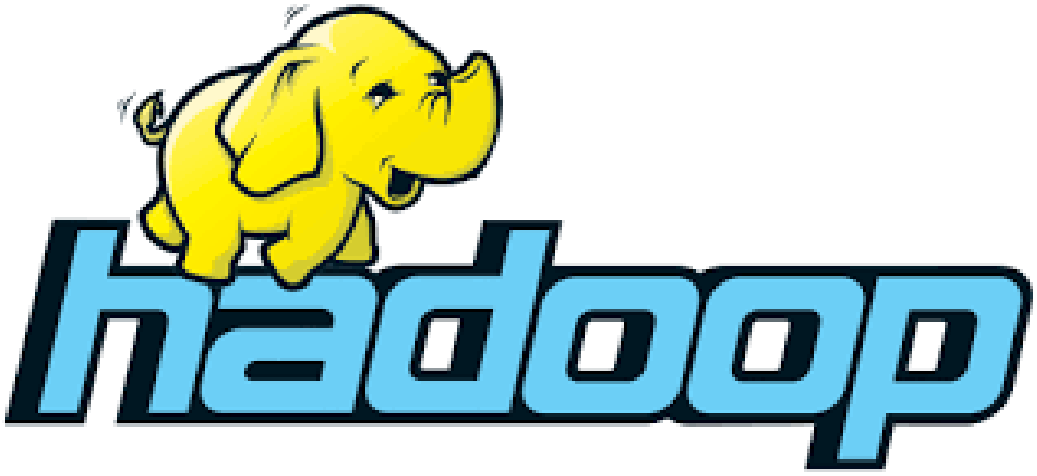


Fuente el Autor

ANEXO B

Manual de configuración e instalación apache Hadoop.

Manual de instalación y configuración



Apache hadoop en alta disponibilidad

Tabla de contenido

Tabla de imágenes

Introducción

Este manual está dirigido para todas las personas que deseen instalar y configurar un clúster con el software apache Hadoop en alta disponibilidad, sé asume que las personas que deseen utilizarlo o implementarlo deben estar familiarizados con conceptos mínimos de sistema operativo Centos7 que es el sistema en el cual está desarrollado este manual.

Se desarrolla sobre una arquitectura en alta disponibilidad en la que se usan 4 nodos esclavos y 2 nodos master, que puede ser modificado de acuerdo a los requerimientos de cada infraestructura.

Conceptos Fundamentales

Características generales:

a. Sistema Operativo: Centos7.
b. Apache Hadoop 2.8.5.
c. Java versión jdk1.8.0_181.

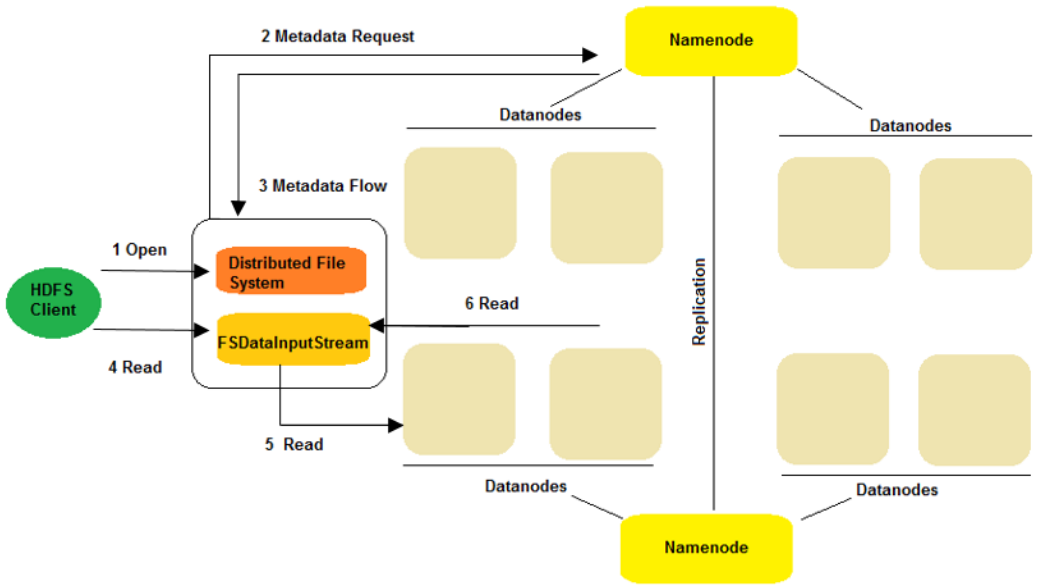
Pre-requisitos generales:

Para el correcto desarrollo de la instalación y configuración se debe tener claro los siguientes aspectos

- Todos los nodos deben tener configurado su host para la comunicación entre ellos.
- Todos los nodos deben tener configurado la misma versión java instaladas, así como su ruta $HOME.
- No configurar apache Hadoop con el usuario root del sistema, con la finalidad de evitar problemas de configuración.

Esquema de infraestructura:

Figura 1. Esquema apache Hadoop pseudo distribuido en alta disponibilidad



Fuente: Autor

Instalación Apache Hadoop

a. Descargar Hadoop

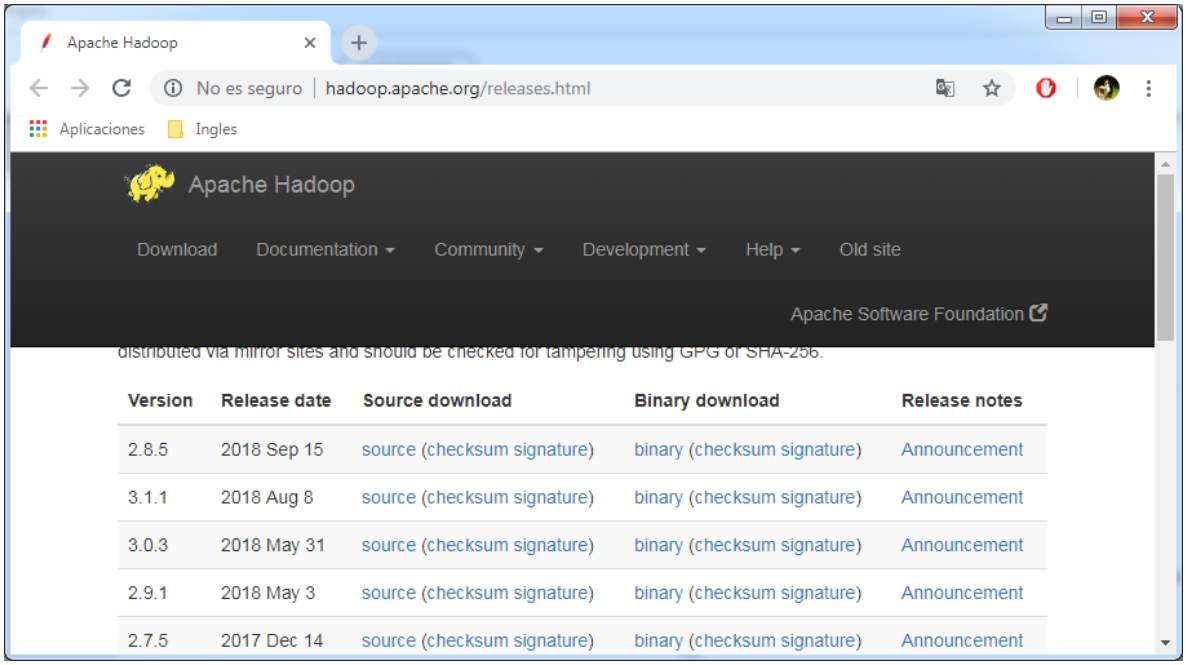
Se procede a descargar de la página oficial el binario de apache Hadoop http://hadoop.apache.org/releases.html en los dos nodos principales no es necesario en los nodos esclavos puesto que la instalación se replicara desde los nodos principales.

Figura 1. Descarga apache Hadoop.

```
[root@hadoop-master ~]# cd Descargas
[root@hadoop-master Descargas]# ll
total 238740
-rw-r--r--. 1 root root 244469481 dic 13  2017 hadoop-2.8.3.tar.gz
[root@hadoop-master Descargas]#
```

Fuente: El autor

Figura 2. Página Oficial de apache Hadoop.



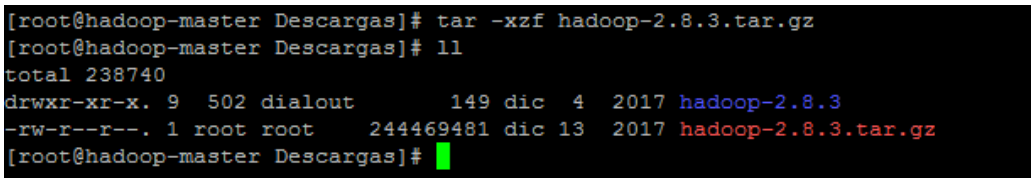
Fuente: El autor

b. Configurar ubicación Hadoop

Una vez descargada la versión de apache en nuestro sistema Linux hadoop-2.X.X.tar.gz procedemos a descomprimirlo usando el comando.

>tar -xzf hadoop-2.8.3.tar.gz

Figura 3. Descomprimir Hadoop



Fuente: El autor

Como buena práctica moveremos la carpeta descomprimida anterior mente en el directorio /usr/local/hadoop/ el cual es una carpeta en la que los diferentes usuarios del sistema podrán tener acceso

>mv hadoop-2.8.3 /usr/local/hadoop/

Crearemos un usuario hadoop con el fin de no manipular hadoop con un usuario root

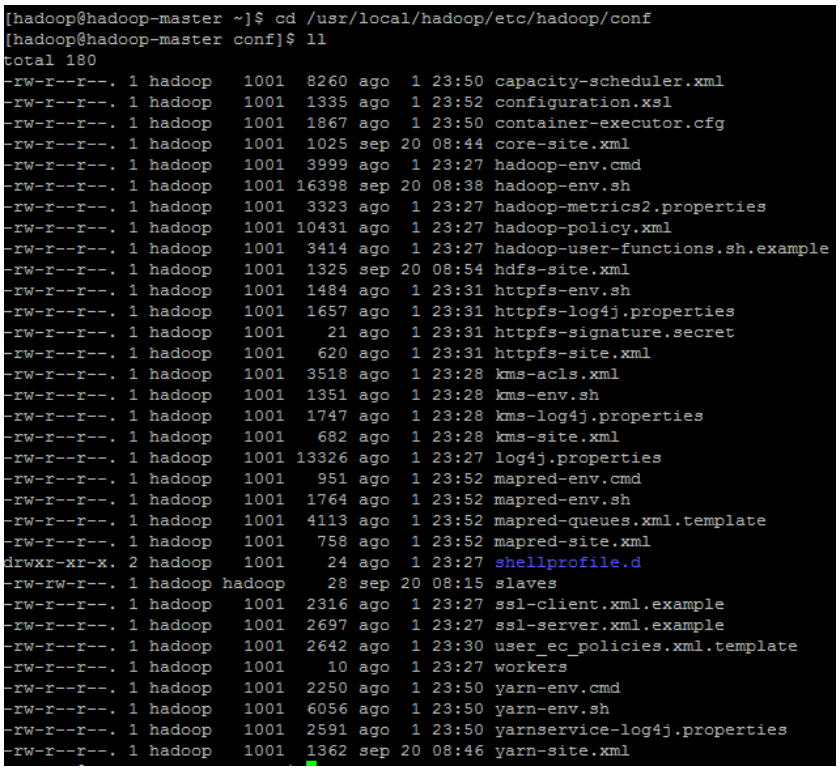
>useradd hadoop
>passwd hadoop

c. Configuración de archivos

En /usr/local/hadoop/etc/hadoop/conf/ encontramos todos los archivos de configuración necesarios, pero para nuestro caso utilizáremos 3 archivos principalmente

>core-site.xml
>hdfs-site.xml
>slaves

Figura 4. Archivos de configuración apache Hadoop



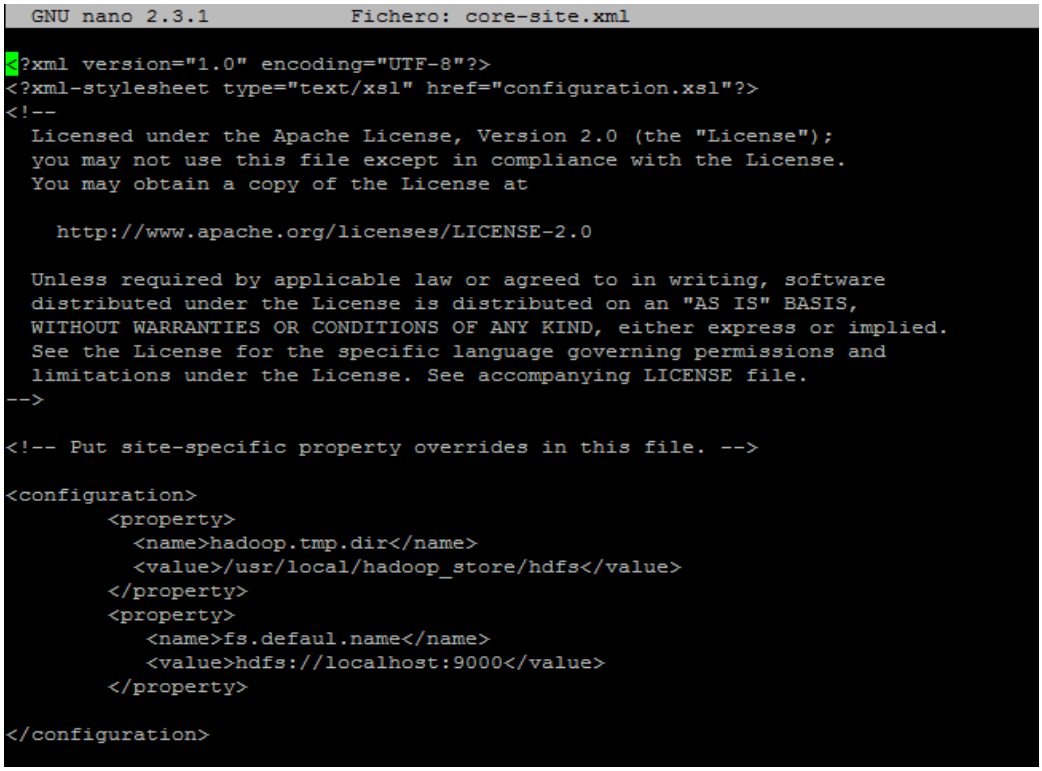
Fuente. El autor

Iniciamos configurando el archivo core-site.xml en el cual tiene dos propiedades fundamentales “hadoop.tmp.dir”,”fs.default.name” con las siguientes líneas de código y guardamos la configuración.

nano core-site.xml

```
<configuration>
                <property>
        <name>hadoop.tmp.dir</name>
        <value>/usr/local/hadoop_store/hdfs</value>
      </property>
      <property>
        <name>fs.defaul.name</name>
        <value>hdfs://localhost:9000</value>
      </property>
</configuration>
```

Figura 5. Archivo de configuración core-site.xml.

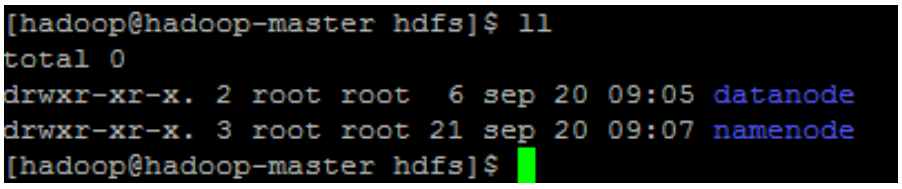


Fuente: El autor

Antes de configurar el archivo “hdfs-site.xml” crearemos dos carpetas en la ruta /usr/local/hadoop_store/hdfs/ una para guardar las configuraciones del namenode y la otra para las configuraciones de datanodes.

>mkdir -p /usr/local/hadoop_store/hdfs/namenode
>mkdir -p /usr/local/hadoop_store/hdfs/datanode

Figura 6. Directorios creados para datanode y namenode.



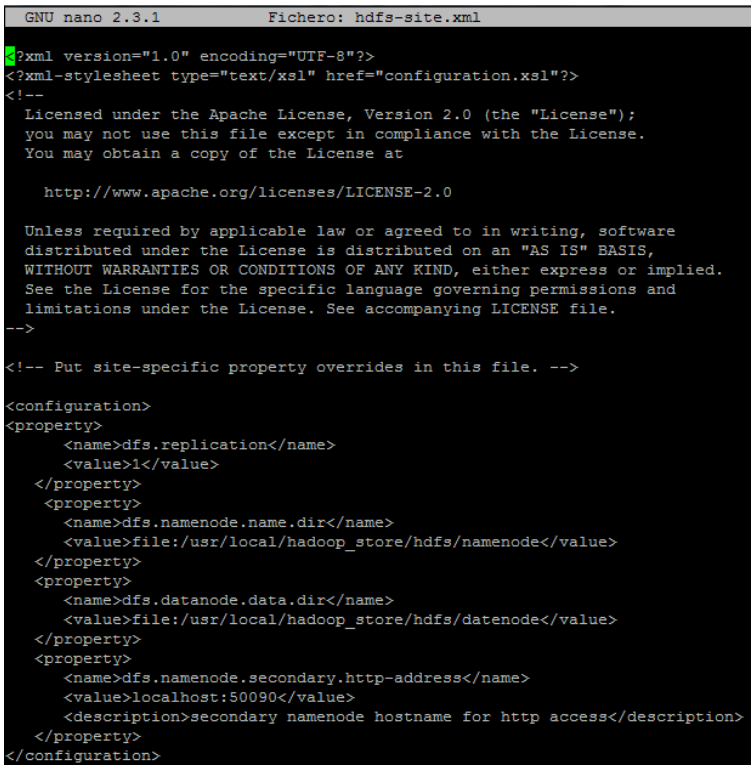
Fuente: El autor

Luego configuramos el archivo “hdfs-site.xml” el cual contiene las característica para determinar el número de iteraciones en que las partes del clúster se replicaran, así como en que parte del file system el datanodes debería guardar sus bloques y configurar el puerto de escuchas en que el Nodo secundario estará escuchando.

nano hdfs-site.xml

```
<Configuration>
    <property>
      <name>dfs.replication</name>
      <value>1</value>
   </property>
    <Property>
      <name>dfs.namenode.name.dir</name>
      <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
   </property>
   <property>
      <name>dfs.datanode.data.dir</name>
      <value>file:/usr/local/hadoop_store/hdfs/datenode</value>
   </property>
   <property>
      <name>dfs.namenode.secondary.http-address</name>
      <value>localhost:50090</value>
      <description>secondary namenode hostname for http access</description>
   </property>
</configuration>
```

Figura 7. Archivo de configuración hdfs-site.xml



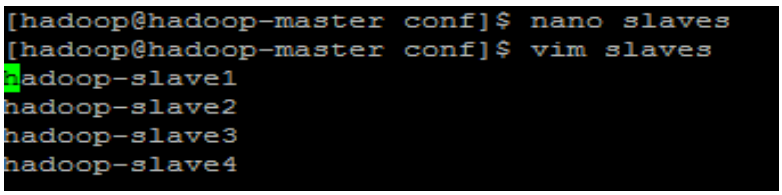
Fuente: El autor

Nuestro siguiente archivo es “slaves” en el cual declaramos los hostnames de las computadoras con el rol de datanodes.

>nano slaves

hadoop-slave1
hadoop-slave2
hadoop-slave3
hadoop-slave4

Figura 8. Archivo de configuración slaves



Fuente: El Autor

d. Configuración de variables de entorno

Una vez configurados los archivos de hadoop procedemos a declarar las variables que necesitaremos para este caso el archivo .bashrc de nuestro usuario hadoop, por esto es necesario crear un usuario distinto a root para no dañar el .bashc de root.

Una vez realizadas estas configuración aplicamos los cambios con el comando source ~/.bashrc

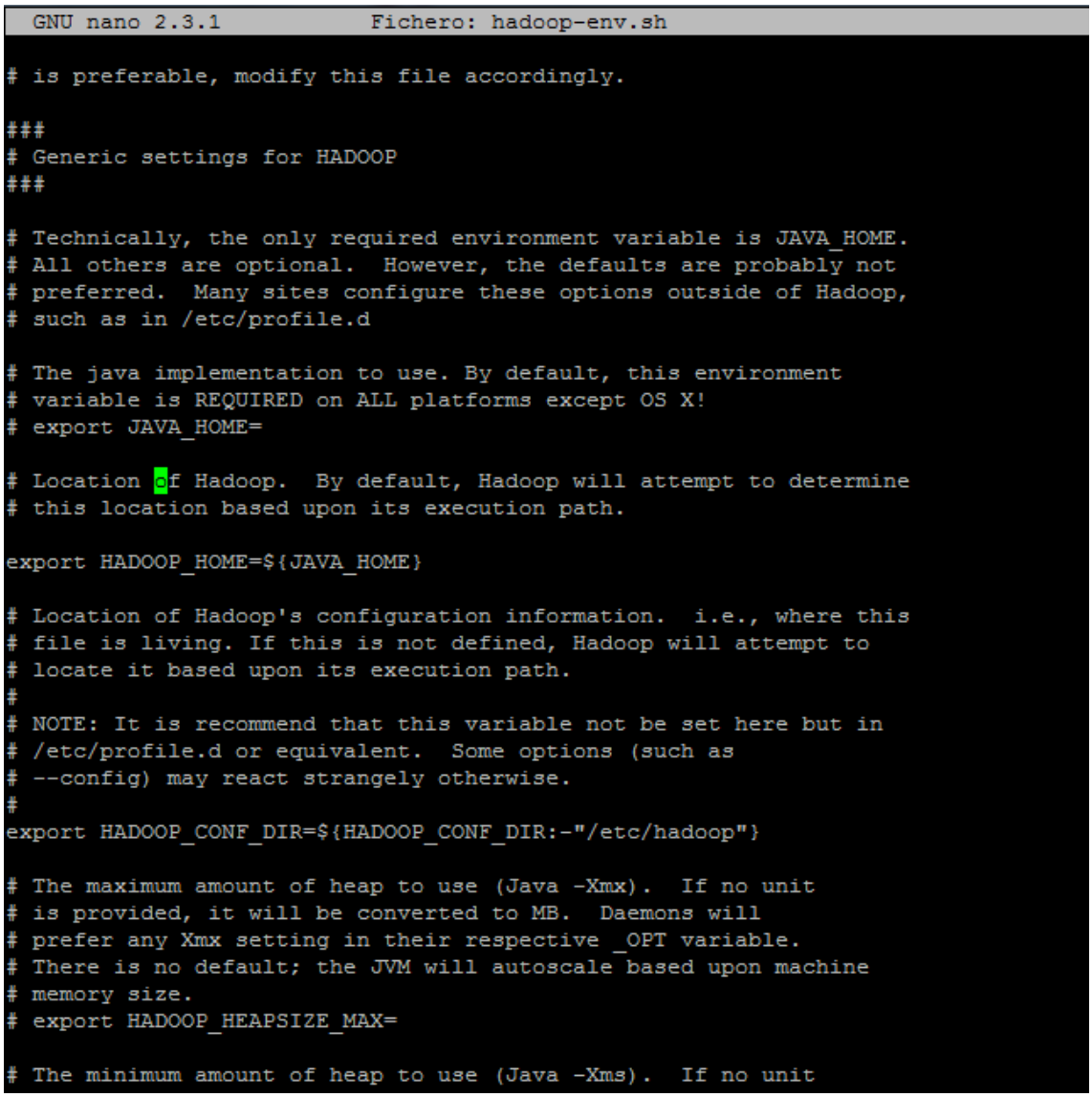
Figura 9. Archivo de configuración .bashc

```
nano ~/.bashrc

export HADOOP_INSTALL=/usr/local/hadoop
export HADOOP_CONF_DIR=$HADOOP_INSTALL/etc/hadoop/conf
export PATH=$PATH:$HADOOP_INSTALL/bin:$HADOOP_CONF_DIR
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-2.b14.el7.x86_64
PATH=$PATH:$JAVA_HOME/bin
export PATH
export HDFS_NAMENODE_USER="root"
export HDFS_DATANODE_USER="root"
export HDFS_SECONDARYNAMENODE_USER="root"
export YARN_RESOURCEMANAGER_USER="root"
export YARN_NODEMANAGER_USER="root"


source ~/.bashrc
```

Fuente: El Autor

Una vez configurado esto nuestro archivo hadoop-env.sh se tuvo que configurar de la siguiente memoria

Figura 10. Archivo de configuración hadoop-env.sh

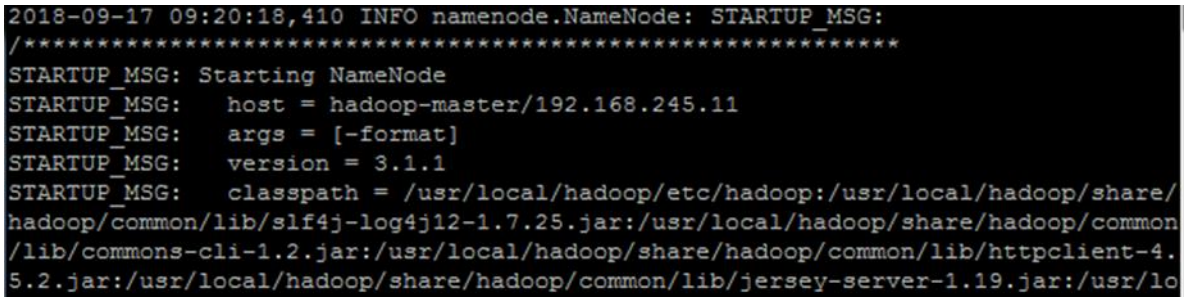


Fuente: El autor

e. Darle formato a nuestro namenode

Utilizando el comando “hdfs namenode –format” desde la ruta “cd /usr/local/hadoop_store/hdfs/” daremos el formato a nuestro namenode obteniendo como resultado lo siguiente

Figura 11. Formato namenodo “hdfs namenode –format”



Fuente: El Autor

f. Configuración de inicio de sesión sin claves

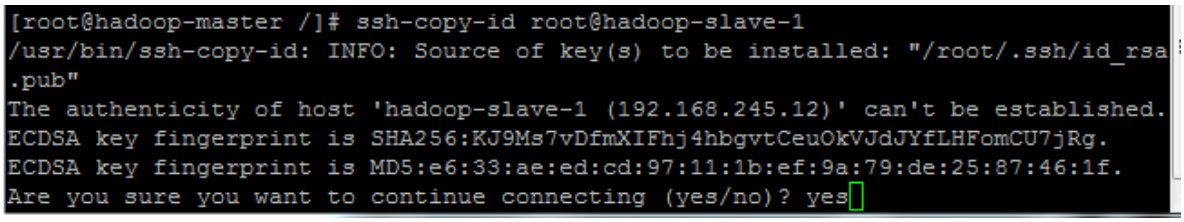
En cada uno de los nodo ejecutamos la siguiente líneas de comando línea por línea para los nodos importante no continuar con la siguiente línea hasta no haberla ejecutado en todos los nodos, esto nos permite tener conexión con todos los nodos.

Figura 12 Inicio de sesión sin claves

```
su
ssh-keygen -t rsa
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop@hadoop-master
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop@hadoop-slave1
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop@hadoop-slave2
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop@hadoop-slave3
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop@hadoop-slave4
chmod 0600 ~/.ssh/authorized_keys
```

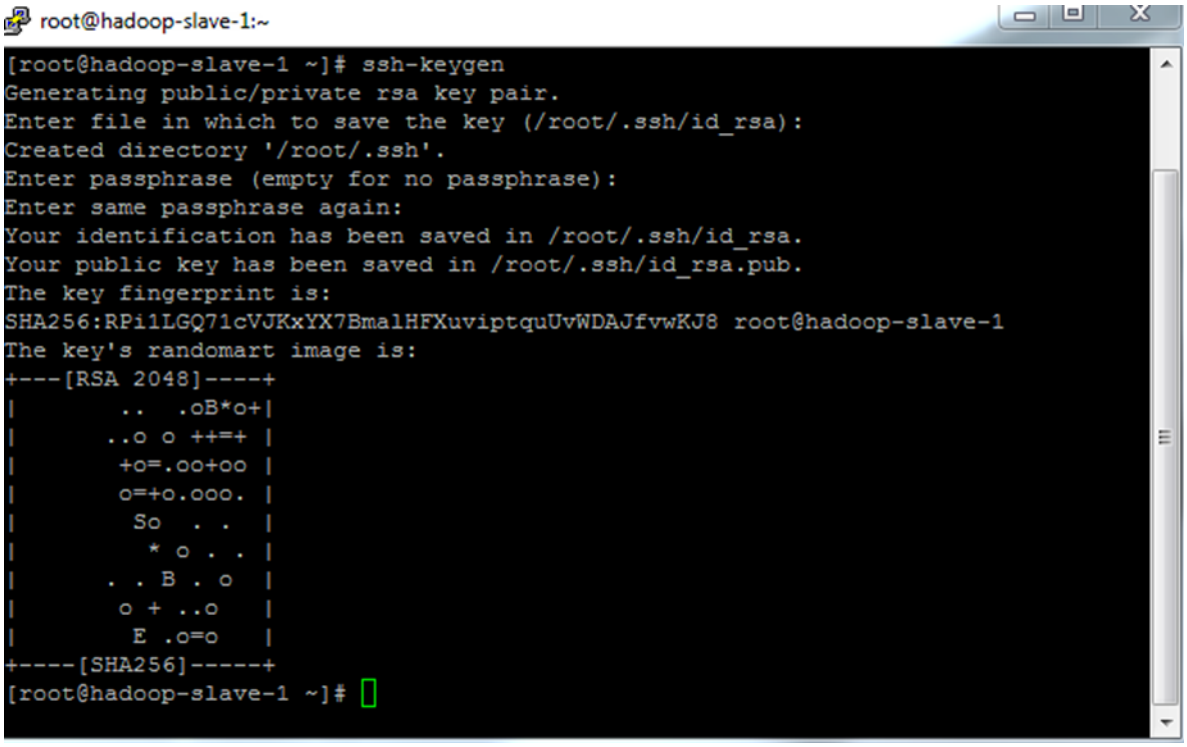
Fuente: El Autor

Figura 13. Conexión hadoop-slave-1



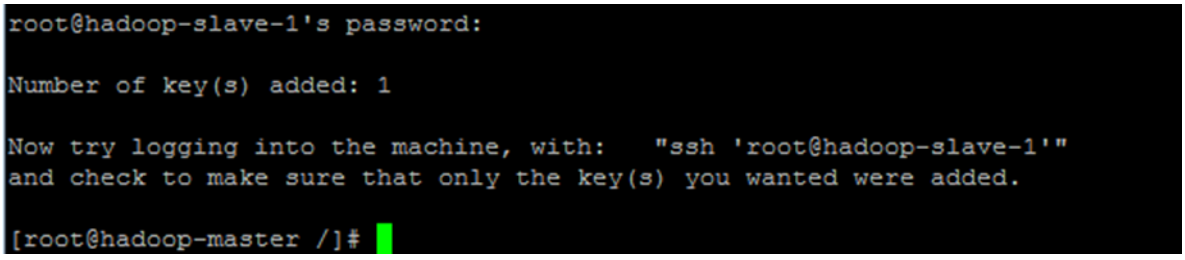
Fuente: El Autor

Figura 14. ssh-keygen



Fuente: El autor

Figura 15 Verificar password slave-1



Fuente: El autor

g. Configuración de Hadoop en nodos esclavos

Los nodos esclavos pueden ser instalados por separado o pueden ser instalados desde el namenode ahorrando tiempo en descargas y configuración que serían innecesarias, por eso desde el nodo maestro ejecutamos estas líneas desde la ruta /usr/local/hadoop/ la cual configura Hadoop en los nodos esclavos.

Figura 16 Configuración de hadoop en nodos.

```
cd /usr/local/hadoop/
scp -r hadoop hadoop-slave1:/usr/local/hadoop/
scp -r hadoop hadoop-slave2:/usr/local/hadoop/
scp -r hadoop hadoop-slave3:/usr/local/hadoop/
scp -r hadoop hadoop-slave4:/usr/local/hadoop/
```

Fuente: El Autor

h. Iniciar servicios

Una vez termina la instalación de Hadoop en los nodos esclavos procedemos a iniciar el servicio y disfrutar de un clúster de apache Hadoop en alta disponibilidad

Figura 17 Iniciar Servicios en Apache Hadoop

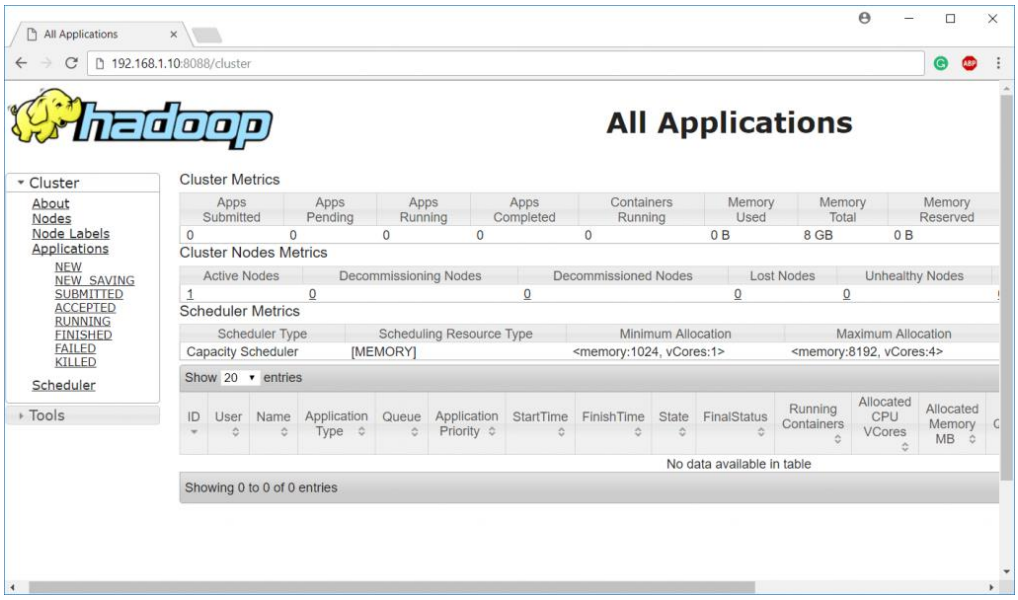
```
cd $HADOOP_HOME/sbin
start-all.sh
```

Fuente: El Autor

i. Verificación de Hadoop en el browser

Cada vez que se agrega un nodo en apache Hadoop lo podemos verificar en el navegador una vez iniciado el servicio.

Figura 18 Prueba de conexión Hadoop



Fuente: El Autor

ANEXO C

Manual de configuración e instalación apache spark.

Manual de instalación y configuración

Apache spark en alta disponibilidad

Tabla de contenido

Tabla de ilustraciones

Introducción

Este manual está dirigido para todas las personas que deseen instalar y configurar el software apache spark en alta disponibilidad siguiendo una arquitectura de chispa maestro /esclavo con dos demonios principales y un nodo administrador de clúster, sé asume que las personas que deseen utilizarlo o implementarlo deben estar familiarizados con conceptos mínimos de sistema operativo Centos7 que es el sistema en el cual está desarrollado este manual.

Se desarrolla sobre una arquitectura en alta disponibilidad en la que se usan 3 nodos para recrear un ambiente clúster (master Daemos-Demonio Trabajador-clúster manager) que puede ser modificado de acuerdo a los requerimientos de cada infraestructura.

Conceptos Fundamentales

Características generales:

a. Sistema Operativo: Centos7.
b. Apache spark.

Pre-requisitos generales:

Para el correcto desarrollo de la instalación y configuración se debe tener claro los siguientes aspectos

- Todos los nodos deben tener configurado su host para la comunicación entre ellos.
- Se debe tener el mismo usuario en todos los nodos, para no tener problemas durante el ssh

Figura 1. Host.

```
192.168.245.11 slave1
192.168.245.12 maestro
192.168.245.13 slave2
```

Fuente: El Autor.

Figura 47.Esquema Apache spark

Fuente: El Autor.

Instalación Apache Spark:

a. Descargar Apache Spark

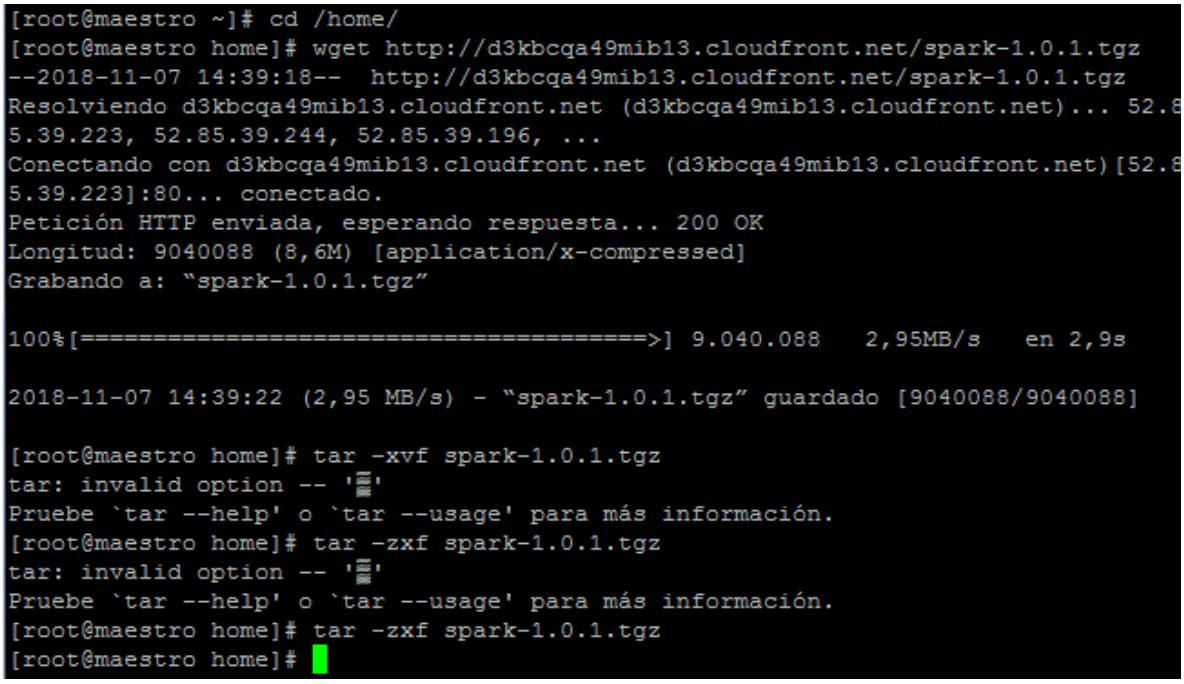
Configuramos el archivo de host de todos los nodos previamente para que no surjan problemas a la hora de la conexión entre ellos y luego descargamos apache spark.

$nano /etc/hosts

192.168.245.11 slave1
192.168.245.12 maestro
192.168.245.13 slave2

cd /home/
wget http://d3kbcqa49mib13.cloudfront.net/spark-1.0.1.tgz
tar –xvf spark-1.0.1.tgz

Figura 2. Descarga Spark



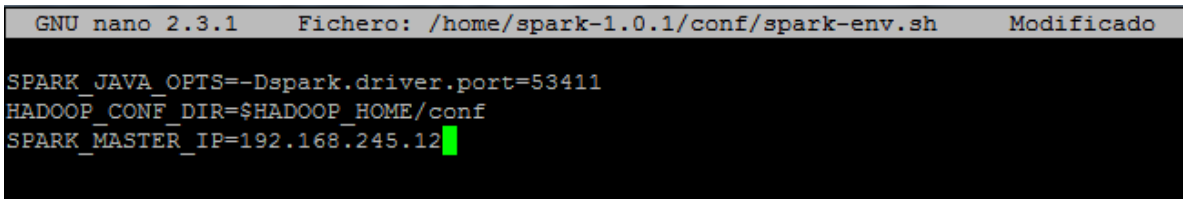
Fuente: El Autor.

b. Configuración de spark-env.sh

En el archivo spark-env.sh configuramos las variables de entorno que nos permiten configurar de forma correcta nuestro clúster, y para ello creamos /home/spark-1.0.1 /conf/spark-env.sh con la ayuda del comando nano

nano /home/spark-1.0.1/conf/spark-env.sh

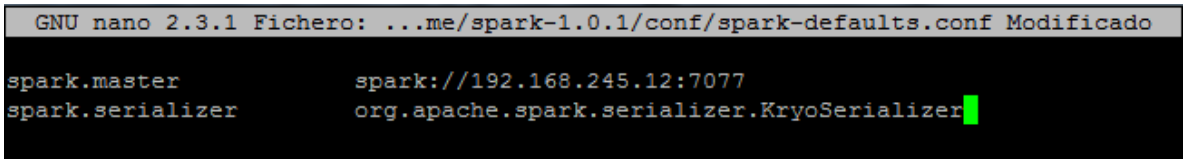
Figura 3. Archivo spark-env.sh



Fuente: El Autor.

nano /home/spark-1.0.1/conf/spark-defaults.conf

Figura 4. Archivo spark-defaults.conf.

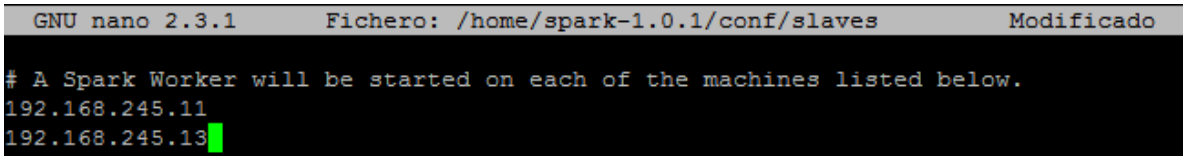


Fuente: El Autor.

Ahora agregaremos el nombre de host de los nodos esclavos en la ruta:

/home/spark-1.0.1 /conf/slaves.

Figura 5. Archivo slaves.



Fuente: El Autor

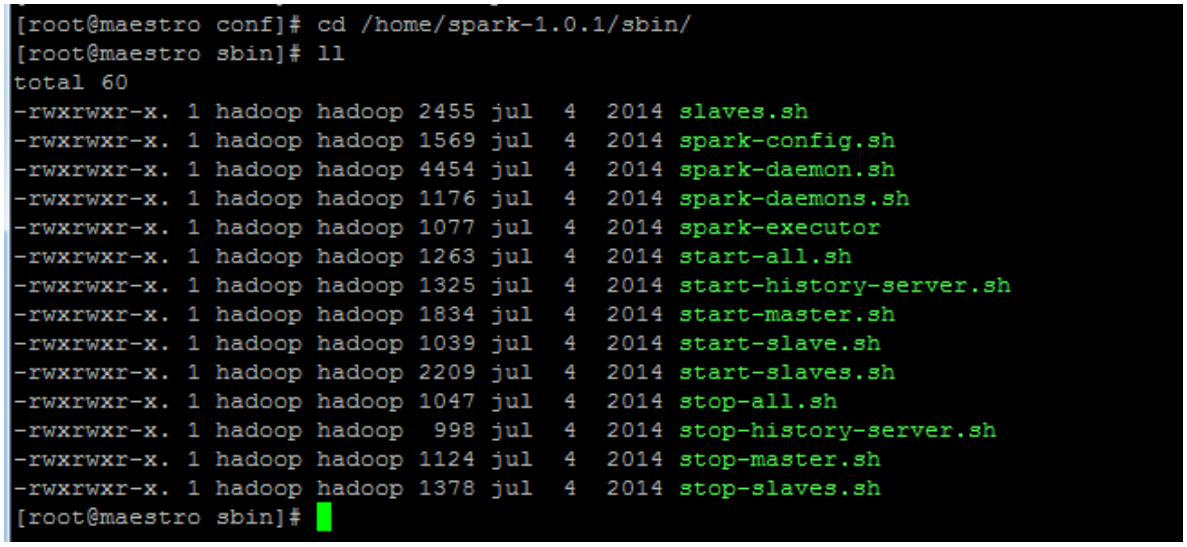
Nota: Estos mismos pasos se repiten en los dos nodos esclavos

c. Iniciar spark

En la tura de configuración de spark encontramos algunas líneas de arranque para las diferentes anatomías que spark usa usaremos el archivo de inicio y stop de spark para que tome los cambios.

```
sh /home/spark-1.0.1/sbin/start-all.sh
sh /home/spark-1.0.1/sbin/stop-all.sh
```

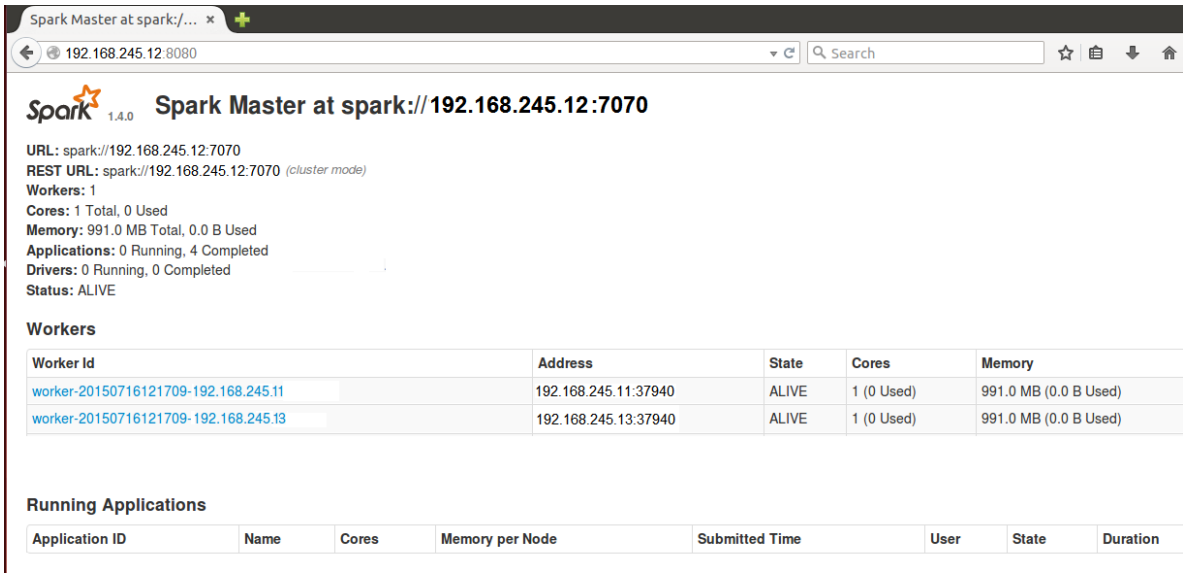
Figura 48.Archivos .sh de spark.



Fuente: El Autor.

Una vez que ejecutamos los pasos en el orden del manual ya queda instalado y configurado nuestro clúster de apache spark en alta disponibilidad ejecutando en el browser de cualquiera de los nodo la ruta http://<ip-maestro>:8080 nos muestra la consola del spark master y el número de nodos que tenemos.

Figura 7. Spark Master



Fuente: El Autor

ANEXO D

Manual de configuración e instalación apache Cassandra.

Manual de instalación y configuración

Apache cassandra en alta disponibilidad

Tabla de contenido

Tabla de ilustraciones

Introducción

Este manual está dirigido para todas las personas que deseen instalar y configurar el software apache Cassandra en alta disponibilidad, sé asume que las personas que deseen utilizarlo o implementarlo deben estar familiarizados con conceptos mínimos de sistema operativo Centos7 que es el sistema en el cual está desarrollado este manual.

Se desarrolla sobre una arquitectura en alta disponibilidad en la que se usan 3 nodos para recrear un ambiente clúster que puede ser modificado de acuerdo a los requerimientos de cada infraestructura.

Conceptos Fundamentales

Características generales:

- Sistema Operativo: Centos7.
- Apache Cassandra 3.11.3

Pre-requisitos generales:

Para el correcto desarrollo de la instalación y configuración se debe tener claro los siguientes aspectos

- Todos los nodos deben tener configurado su host para la comunicación entre ellos.

Figura 49.Valores host.

Fuente: El autor.

Instalación Apache Cassandra:

c. **Preparar comunicación**

Configuramos el archivo de host de todos los nodos previamente para que no surjan problemas a la hora de la conexión entre ellos.

$nano /etc/hosts

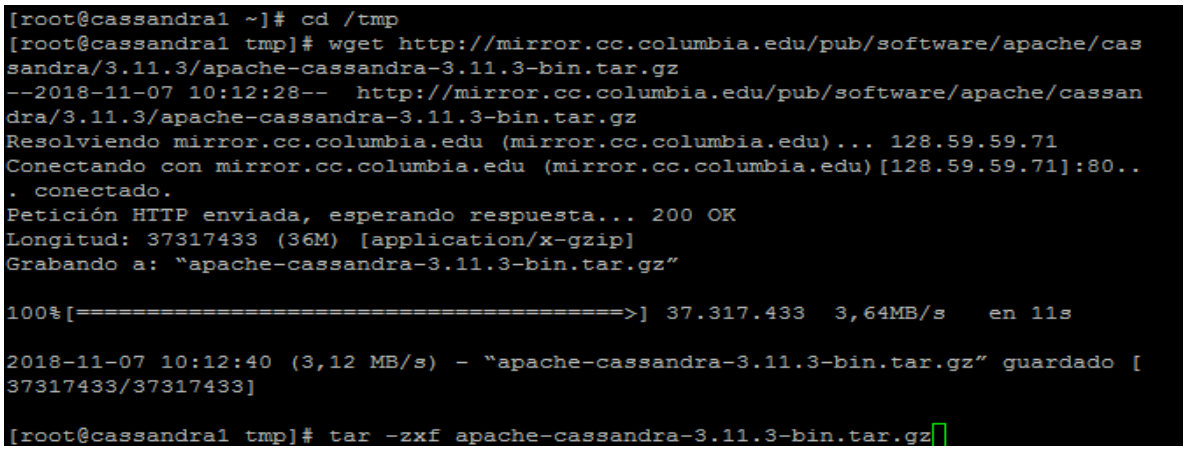
192.168.245.11 cassandra1
192.168.245.12 cassandra2
192.168.245.13 cassandra3

d. Descargar cassandra

La instalación de cassandra se realizara en el carpeta tmp como buena práctica para que todos los usuarios tengan acceso a ella y una vez descargado el apache cassandra se procede a descomprimir en la misma ruta tmp con el comando tar -zxf.

```
cd /tmp

wget http://mirror.cc.columbia.edu/pub/software/apache/cassandra/3.11.3/apache-cassandra-3.11.3-bin.tar.gz
tar -zxf apache-cassandra-3.11.3-bin.tar.gz
```

Figura 50. Wget apache-cassandra

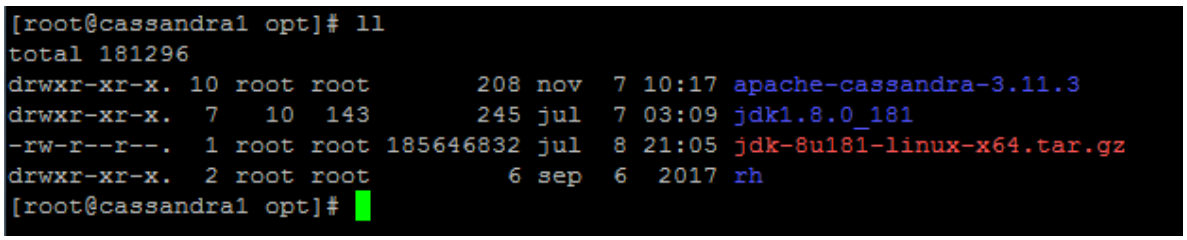


Fuente: El Autor

Ahora moveremos el apache-cassandra a la ruta

```
sudo mv apache-cassandra-3.11.3 / / opt /
```

Figura 51.Apache-cassandra en la carpeta opt.



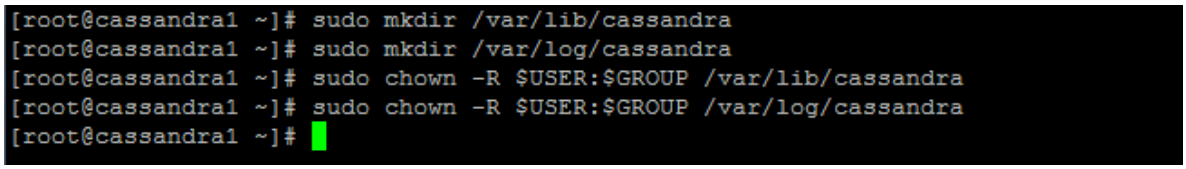
Fuente: El Autor.

e. Verificar acceso a carpetas de registro

Creamos las carpetas cassandra en las carpetas de registro de nuestro sistema operativo con el fin de que se pueda escribir en ella.

```
sudo mkdir /var/lib/cassandra
sudo mkdir /var/log/cassandra
sudo chown -R $USER:$GROUP /var/lib/cassandra
sudo chown -R $USER:$GROUP /var/log/cassandra
```

Figura 52. Carpetas de registro cassandra.



Fuente: El autor.

Configuramos la variable de entorno en al ruta /etc/profile.d/cassandra.sh usando el comando cat

```
vi /etc/profile.d/cassandra.sh
export CASSANDRA_HOME=/opt/apache-cassandra-3.11.3
export PATH=$PATH:$CASSANDRA_HOME/bin
```

Reiniciamos el nodo para que los cambios surjan efecto

```
reboot
```

f. Configuración de clúster Cassandra

Antes de continuar con la configuración borramos log de cassandra

*sudo rm -rf /var/lib/cassandra/**

Como este manual es para un servicio en alta disponibilidad tendremos 3 nodos un nodo 3 y un solo centro de datos con una sola semilla clúster.

Cassandra1
Cassandra2
Cassandra3

Ahora calcularemos el código para los nodos del clúster con un solo centro de datos con 3 nodos.

*$python -c 'number_of_tokens=3; print [str(((2**64 / number_of_tokens) * i) - 2**63) for i in range(number_of_tokens)]'*

Figura 53. Token

```
[root@cassandra1 ~]# sudo rm -rf /var/lib/cassandra/*
[root@cassandra1 ~]# python -c 'number_of_tokens=3; print [str(((2**64 / number_
f_tokens) * i) - 2**63) for i in range(number_of_tokens)]'
['-9223372036854775808', '-3074457345618258603', '3074457345618258602']
[root@cassandra1 ~]#
```

Fuente:El Autor

Ahora configuramos el archvi yarm de cada uno con de los nodos con los atributos que corresponda modificando cada línea de texto que corresponda.

vi $CASSANDRA_HOME/conf/cassandra.yaml

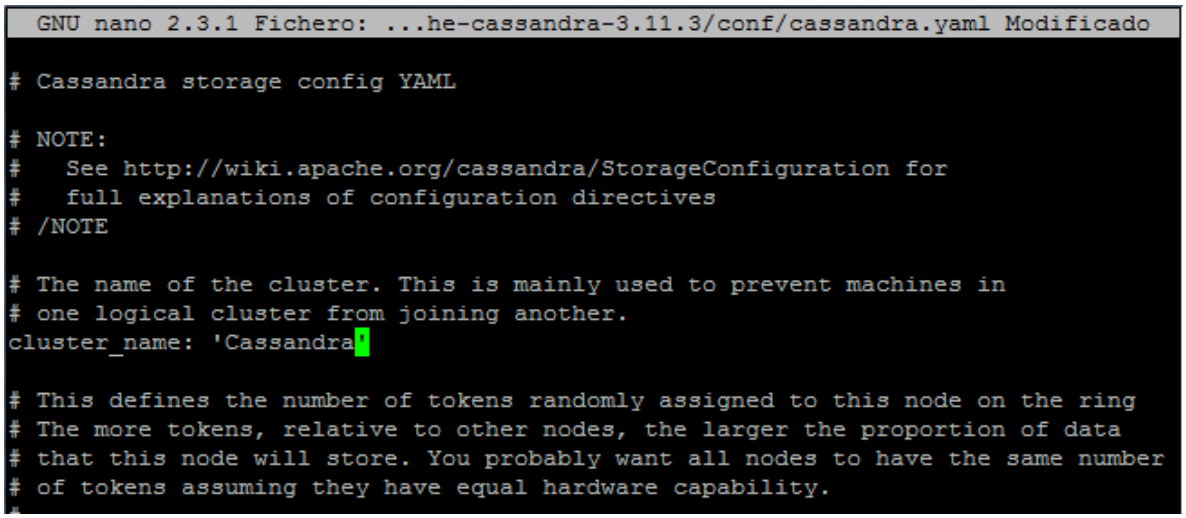
cluster_name: 'Cassandra'
seed_provider:
- seeds: "cassandra1"
rpc_address: 0.0.0.0
endpoint_snitch: SimpleSnitch

initial_token: -9223372036854775808
listen_address: localhost
broadcast_rpc_address: localhost

initial_token: -3074457345618258603
listen_address: localhost
broadcast_rpc_address: localhost

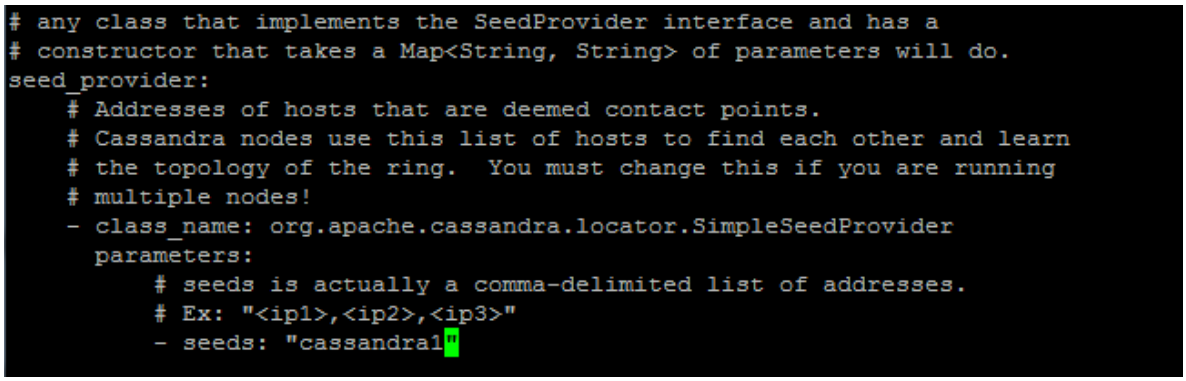
initial_token: 3074457345618258602
listen_address: localhost
broadcast_rpc_address: localhost

Figura 54. Cassandra.yaml cluster name



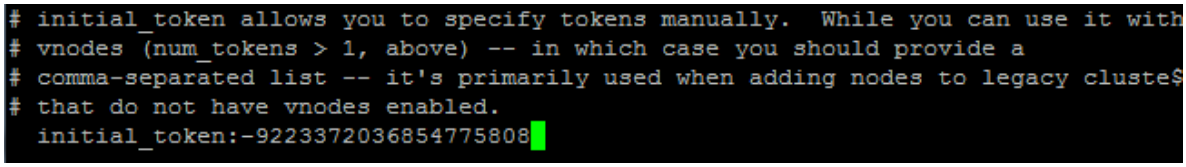
Fuente: El Autor.

Figura 55.Cassandra.yaml seeds.



Fuente: El Autor.

Figura 56.Cassandra.yaml Token.



Fuente: El Autor.

Ya configurados cassandra.yaml en todos los nodos iniciamos cassandra en el nodo semilla

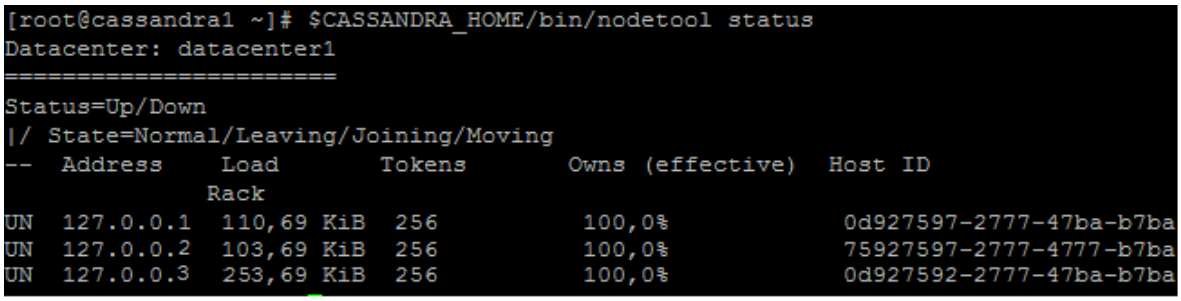
```
sudo sh $CASSANDRA_HOME/bin/cassandra
```

g. Prueba de conexión

Ahora verificaremos que todos los nodos estén conectados con los comandos y tenemos nuestro clúster de apache cassandra en alta disponibilidad

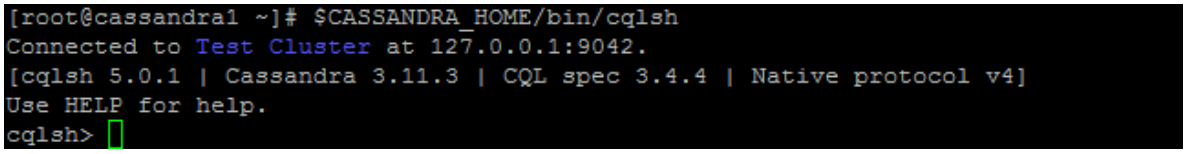
```
$CASSANDRA_HOME/bin/nodetool describecluster
$CASSANDRA_HOME/bin/nodetool status
```

Figura 57. prueba de conexión.



Fuente: El Autor.

Figura 58.Prueba Test Clúster.



Fuente: El Autor

ANEXO E

Manual de configuración e instalación MogoDB.

Manual de instalación y configuración

MongoDB en alta disponibilidad

Tabla de contenido

Tabla de ilustraciones

Introducción

Este manual está dirigido para todas las personas que deseen instalar y configurar el software mongoDB en alta disponibilidad, sé asume que las personas que deseen utilizarlo o implementarlo deben estar familiarizados con conceptos mínimos de sistema operativo Centos7 que es el sistema en el cual está desarrollado este manual.

Se desarrolla sobre una arquitectura en alta disponibilidad en la que se usan 3 nodos para recrear un ambiente clúster que puede ser modificado de acuerdo a los requerimientos de cada infraestructura.

Conceptos Fundamentales

Características generales:

h. **Sistema Operativo: Centos7.**
i. **MongoDB-ORG-3.2**

Pre-requisitos generales:

Para el correcto desarrollo de la instalación y configuración se debe tener claro los siguientes aspectos

- Todos los nodos deben tener configurado su host para la comunicación entre ellos.

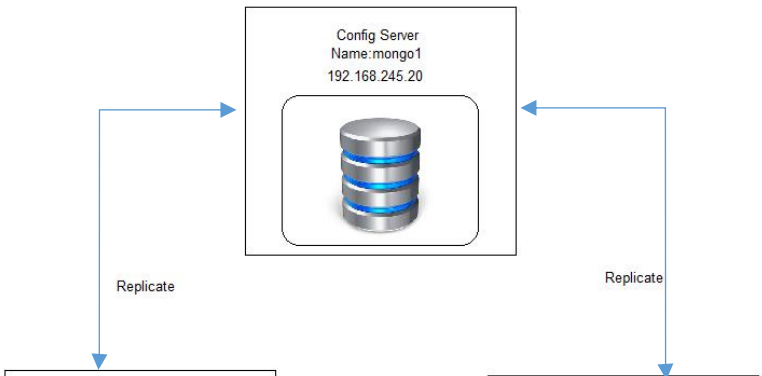
Figura 59.Valores host.

```
192.168.245.20 mongo1
192.168.245.21 mongo2
192.168.245.22 mongo3
```

Fuente: El autor.

Esquema de infraestructura:

.Figura 60. Esquema mongoDB en alta disponibilidad.



Fuente: El autor .

Instalación MongoDB:

j. **Preparar comunicación**

Configuramos el archivo de host de todos los nodos previamente para que no surjan problemas a la hora de la conexión entre ellos.

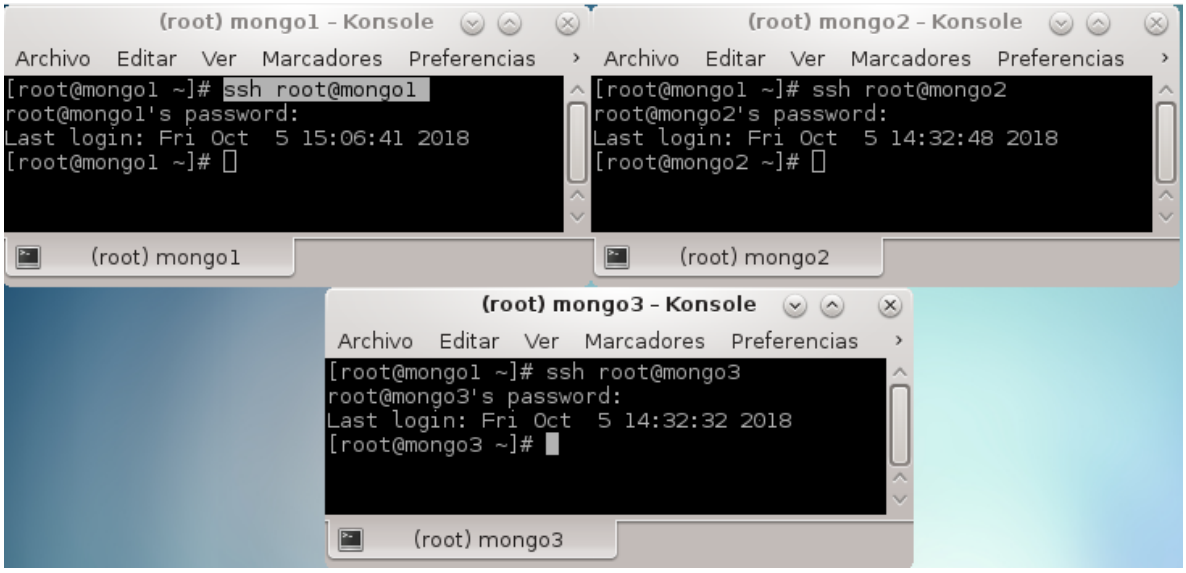
```
$nano /etc/hosts
```

```
192.168.245.15 mongo1
192.168.245.15 mongo2
192.168.245.15 mongo3
```

Antes de iniciar con la instalación de mongoDB en alta disponibilidad tenemos que tener conexión entre los nodos y el servidor con ssh.

```
ssh root@mongo1
ssh root@mongo3
ssh root@mongo3
```

Figura 61. Conexión entre nodos.



Fuente: El autor.

Deshabilitaremos el módulo de seguridad SElinux en la fila 7 de la ruta /etc/sysconfig/selinux de todos los servidores.

Figura 62. SElinux.

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of three two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected proc
e protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

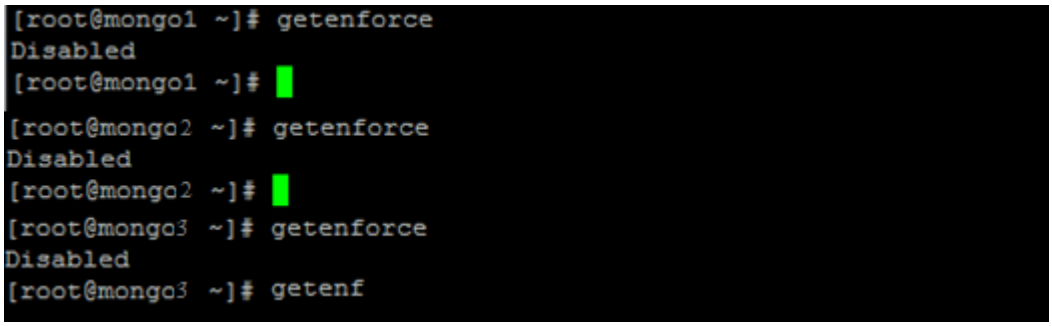
Fuente: El autor.

Guardamos y reiniciamos los servidores

$reboot

Verificamos que los cambios se efectuaron

Figura 63. Cambios Exitosos getenforce.



Fuente. El Autor.

k. **Instalación de repositorio MongoDB en todos los nodos**

Agregaremos el repositorio de mongoDB "mongodb.repo" en la ruta /etc/yum.repos.d en todos los nodos antes de la instalación del paquete de mongodb-org

Nota: si el directorio yum.repos.d no está creado, el paso siguiente será crearlo.

Figura 64. Repositorio mongodb.repo.

Fuente: El autor.

l. **Configuración de Firewalld**

Al inicio de la configuración de mongoDB deshabilitamos SELinux ahora tenemos que habilitar en todos los nodos los puertos de MongoDB y SSH así como el firewalld, instalamos firewalld con el comando:

Yum –y install firewalld

Una vez completada la instalación se procede a iniciar y habilitar el firewalld

systemctl start firewalld
systemctl habilita firewalld

Ya habilitado el firewalld se procede a abrir los puertos predeterminados para la comunicación entre nodos “22” y el puerto el MongoDB “27017”, volvemos a iniciar el firewalld para que los cambios sean tomados exitosamente.

firewall-cmd --permanent --add-port=22/tcp
firewall-cmd --permanent --add-port=27017/tcp
firewall-cmd --reload

Figura 65. Configuración firewall-cmd en cada nodo MongoDB.

```
[root@mongo1 ~]# firewall-cmd --permanent --add-port=22/tcp
success
[root@mongo1 ~]# firewall-cmd --permanent --add-port=27017/tcp
success
[root@mongo1 ~]# firewall-cmd --reload
success
[root@mongo2 ~]# firewall-cmd --permanent --add-port=22/tcp
success
[root@mongo2 ~]# firewall-cmd --permanent --add-port=27017/tcp
success
[root@mongo2 ~]# firewall-cmd --reload
success
[root@mongo3 ~]# firewall-cmd --permanent --add-port=22/tcp
success
[root@mongo3 ~]# firewall-cmd --permanent --add-port=27017/tcp
success
[root@mongo3 ~]# firewall-cmd --reload
success
[root@mongo3 ~]#
```

Fuente: El autor.

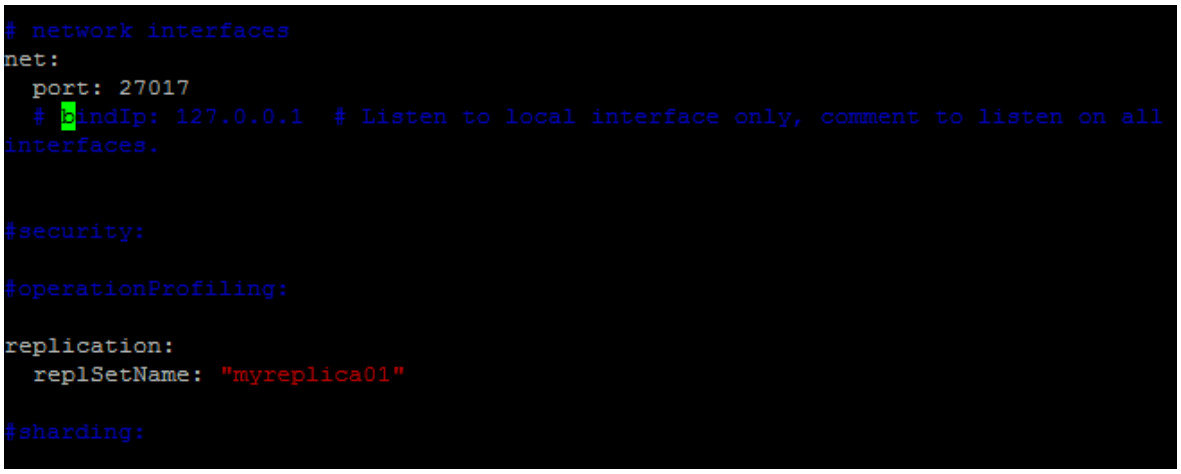
m. Réplicas de MongoDB

Con las réplicas de mongo proporcionaremos alta disponibilidad y tolerancia a fallos para nuestras bases de datos, se compone de nodos primarios y secundarios en los que los nodos primaros son los únicos que escriben y los secundarios tienen sus réplicas.

Configuramos el archivo Mongod.conf y en la línea 27 comentamos 'bindIP',y eliminamos el comentario de la línea 36 estableciendo el nombre de 'myreplica01'.

vim /etc/mongod.conf

Figura 66. Mongod.conf.



Fuente: El autor.

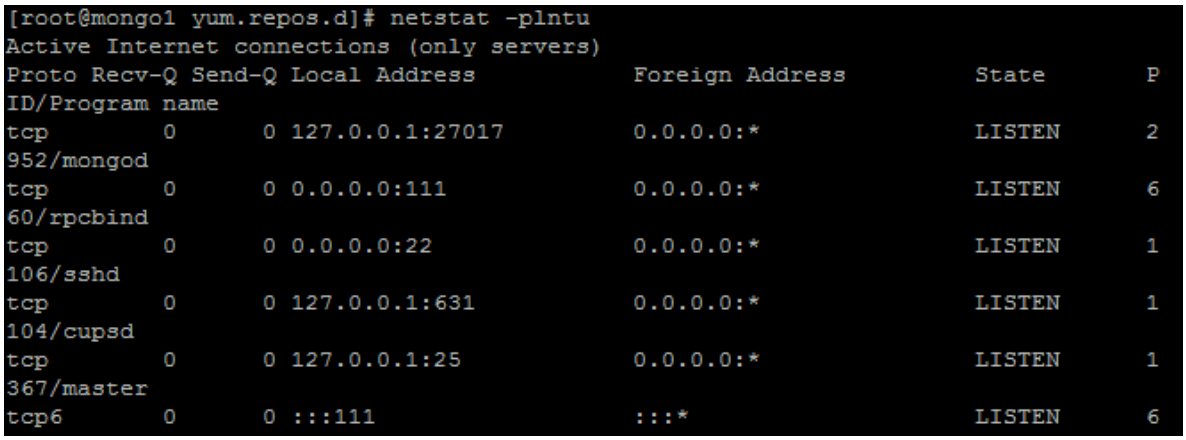
Guardamos el archivo de configuracion y reinicamos Mongod en todos los nodos

```
systemctl restart mongod
```

Comprobar que mongoDB se está ejecutando sobre el servidor ipaddrees y no local host ipaddrees

```
netstat -plntu
```

Figura 67. Netstat –plntu.



Fuente: El autor.

n. Inicializar Réplicas de MongoDB

Crearemos el conjunto de réplicas iniciando el servidor ‘mongo1’

```
ssh root @ mongo1
mongo
```

Iniciamos las réplicas con el siguiente comando y agregamos los nodos de réplicas cerciorándonos que el valor ‘ok’ sea de 1

```
rs.initiate ()
rs.add ("mongo2")
rs.add ("mongo3")
```

Figura 68. Rs.add.

```
myreplica01:OTHER> rs.add("mongo2")
{ "ok" : 1 }
myreplica01:PRIMARY> rs.add("mongo3")
{ "ok" : 1 }
```

Fuente: El autor.

Verificamos el conjunto de replicas

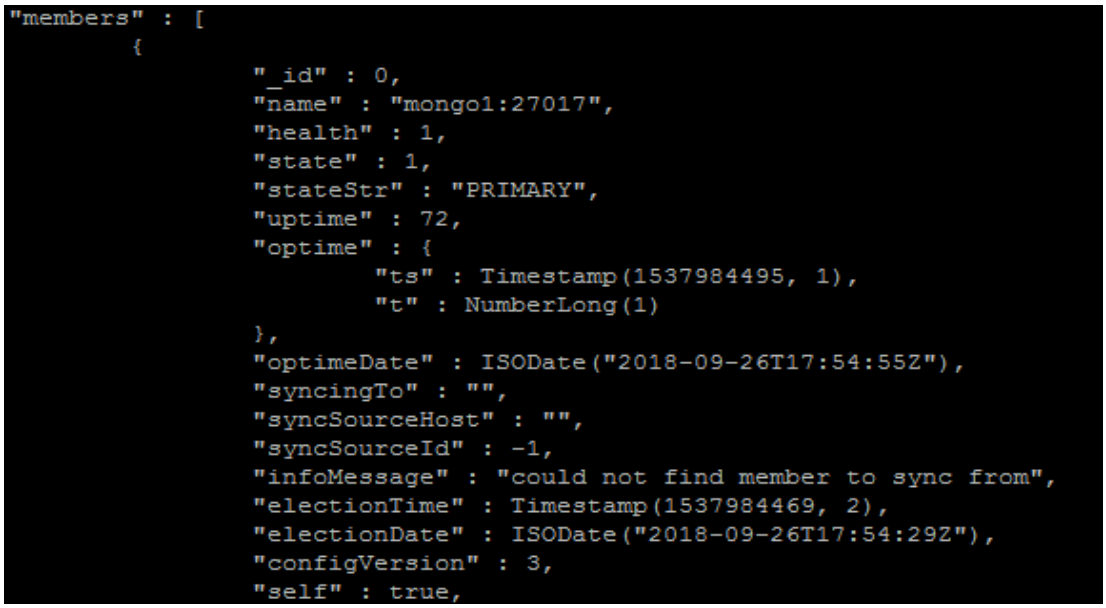
```
rs.status ()
```

Figura 69. Rs.status myreplica01.

```
myreplica01:PRIMARY> rs.status()
{
        "set" : "myreplica01",
        "date" : ISODate("2018-09-26T17:55:04.382Z"),
        "myState" : 1,
        "term" : NumberLong(1),
        "syncingTo" : "",
        "syncSourceHost" : "",
        "syncSourceId" : -1,
        "heartbeatIntervalMillis" : NumberLong(2000),
        "optimes" : {
                "lastCommittedOpTime" : {
                        "ts" : Timestamp(1537984495, 1),
                        "t" : NumberLong(1)
                },
                "appliedOpTime" : {
                        "ts" : Timestamp(1537984495, 1),
                        "t" : NumberLong(1)
                },
                "durableOpTime" : {
                        "ts" : Timestamp(1537984495, 1),
                        "t" : NumberLong(1)
```

Fuente: El Autor.

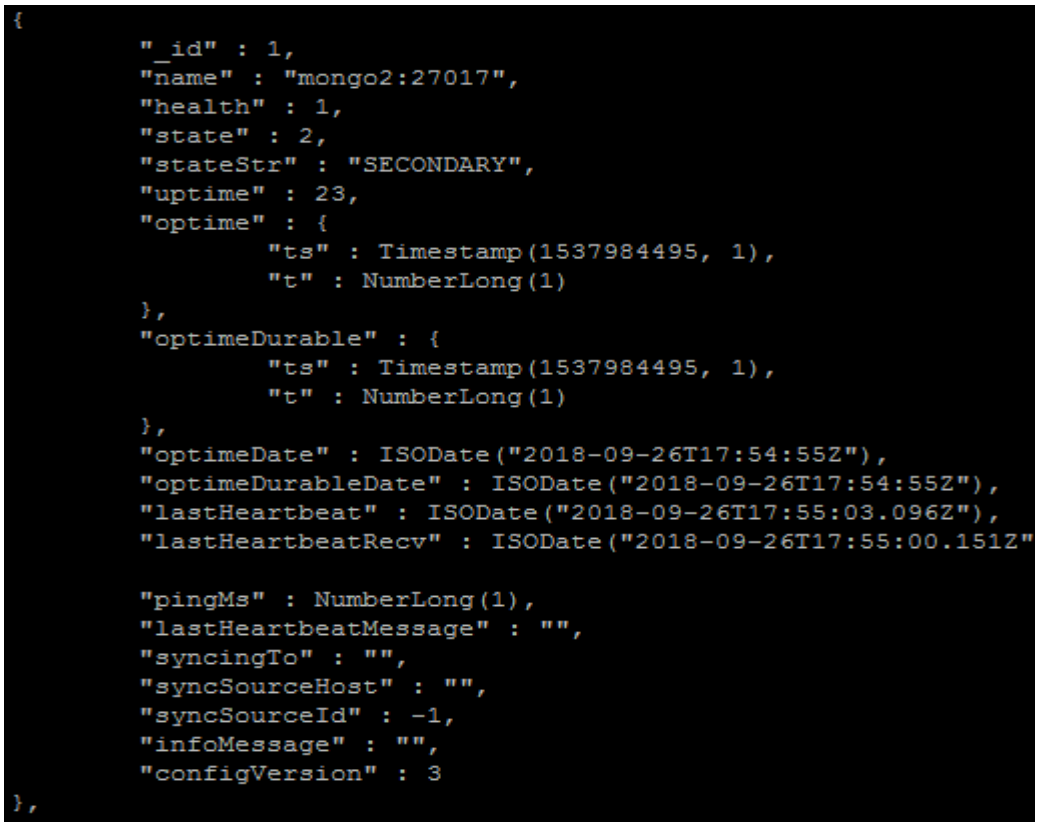
Figura 70. Rs.status mongo1.



```
"members" : [
        {
                "_id" : 0,
                "name" : "mongo1:27017",
                "health" : 1,
                "state" : 1,
                "stateStr" : "PRIMARY",
                "uptime" : 72,
                "optime" : {
                        "ts" : Timestamp(1537984495, 1),
                        "t" : NumberLong(1)
                },
                "optimeDate" : ISODate("2018-09-26T17:54:55Z"),
                "syncingTo" : "",
                "syncSourceHost" : "",
                "syncSourceId" : -1,
                "infoMessage" : "could not find member to sync from",
                "electionTime" : Timestamp(1537984469, 2),
                "electionDate" : ISODate("2018-09-26T17:54:29Z"),
                "configVersion" : 3,
                "self" : true,
```

Fuente: El Autor.

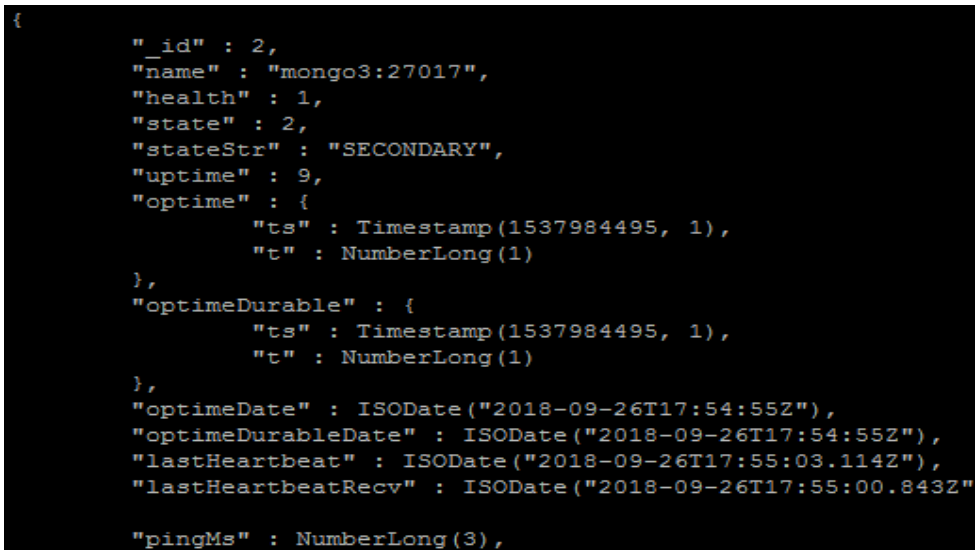
Figura 71. Rs.status mongo2.



```
{
        "_id" : 1,
        "name" : "mongo2:27017",
        "health" : 1,
        "state" : 2,
        "stateStr" : "SECONDARY",
        "uptime" : 23,
        "optime" : {
                "ts" : Timestamp(1537984495, 1),
                "t" : NumberLong(1)
        },
        "optimeDurable" : {
                "ts" : Timestamp(1537984495, 1),
                "t" : NumberLong(1)
        },
        "optimeDate" : ISODate("2018-09-26T17:54:55Z"),
        "optimeDurableDate" : ISODate("2018-09-26T17:54:55Z"),
        "lastHeartbeat" : ISODate("2018-09-26T17:55:03.096Z"),
        "lastHeartbeatRecv" : ISODate("2018-09-26T17:55:00.151Z"

        "pingMs" : NumberLong(1),
        "lastHeartbeatMessage" : "",
        "syncingTo" : "",
        "syncSourceHost" : "",
        "syncSourceId" : -1,
        "infoMessage" : "",
        "configVersion" : 3
},
```

Fuente: El Autor.

Figura 72. Rs.status mongo3.

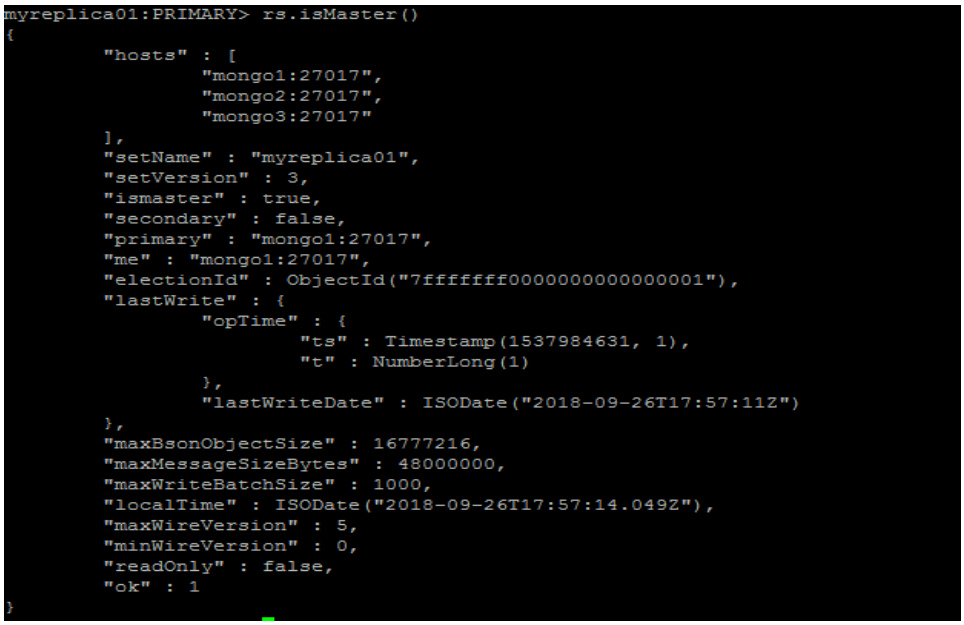


Fuente: El Autor.

Una consulta más resumida seria:

```
rs.isMaster ()
```

Figura 73. Rs.isMaster.



Fuente: El Autor.

o. Prueba de replicación

Una vez agregados los nodos y configurados probamos nuestra replicación de mongo y con esto nos cercioramos que nuestro clúster mongoDB este corriendo a la perfección las diferentes configuración de arquitectura ya depende de los requisitos de cada compañía, crearemos una base de datos con una pequeña información y verificamos que los datos se hayan replicado en los diferentes nodos.

Iniciar sesión en el servidor mongo

```
ssh root @ mongo1
mongo
```

Procedemos a crear una nueva base de datos y una colección de datos

```
use lemp
db.stack.save (
```

```
{
    "desc": "LEMP Stack",
    "apps": ["Linux", "Nginx", "MySQL", "PHP"],
})
```

Figura 74. Nueva base de datos

:

```
myreplica01:PRIMARY> use lemp
switched to db lemp
myreplica01:PRIMARY> db.stack.save (
... {
... "desc": "LEMP Stack",
...     "aplicaciones": ["Linux", "Nginx", "MySQL", "PHP"],
... }
... )
WriteResult({ "nInserted" : 1 })
```

Fuente: El Autor

Ahora ingresamos al Shell de mongo desde el nodo y verificaremos que exista la base de datos anteriormente creada

```
rs.slaveOk ()
show dbs
db.stack.find ()
```

Figura 75. Rs.slaveOK

```
myreplica01:SECONDARY> rs.slaveOk ()
myreplica01:SECONDARY> show dbs
admin  0.000GB
lemp   0.000GB
local  0.000GB
myreplica01:SECONDARY>
```

Fuente: El Autor

Si todo se instaló y configuro correctamente ya tenemos nuestro mongoDB en alta disponibilidad.

ANEXO F
Documento técnico SCRIPTS.

```
#configurar adaptador puente
systemctl stop networkmanager
systemctl disable networkmanager

#configuracion del host para el master y los nodos
su - root
vi /etc/hosts

192.168.245.10 hadoop-master
192.168.245.11 hadoop-master2
192.168.245.12 hadoop-slave1
192.168.245.13 hadoop-slave2
192.168.245.14 hadoop-slave3
192.168.245.15 hadoop-slave4

#instalacionjava para los equipos del cluster


wget --continue --no-check-certificate -O jdk-8u181-linux-x64.tar.gz --header
"Cookie: oraclelicense=a" http://download.oracle.com/otn-pub/java/jdk/8u181-
b13/96a7b8442fe848ef90c96a2fad6ed6d1/jdk-8u181-linux-x64.tar.gz
tar xzf jdk-8u181-linux-x64.tar.gz
cd jdk1.8.0_181/
alternative --install /usr/bin/java java /opt/jdk1.8.0_181/bin/java 2
alternative --config java

#Usuario hadoop
useradd hadoop
passwd hadoop

#instalar hadoop

cd Descargas
wget http://archive.apache.org/dist/hadoop/common/hadoop-2.8.3/hadoop-
2.8.3.tar.gz
tar -xzf hadoop-2.8.3.tar.gz
mv hadoop-2.8.3 /usr/local/hadoop/
cd /usr/local/hadoop/etc/hadoop/conf

nano core-site.xml
<configuration>
                <property>
        <name>hadoop.tmp.dir</name>
```

```
        <value>/usr/local/hadoop_store/hdfs</value>
      </property>
      <property>
        <name>fs.defaul.name</name>
        <value>hdfs://localhost:9000</value>
      </property>
</configuration>

mkdir -p /usr/local/hadoop_store/hdfs/namenode
mkdir -p /usr/local/hadoop_store/hdfs/datanode

nano hdfs-site.xml

<configuration>
   <property>
     <name>dfs.replication</name>
     <value>1</value>
  </property>
   <property>
     <name>dfs.namenode.name.dir</name>
     <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
  </property>
  <property>
     <name>dfs.datanode.data.dir</name>
     <value>file:/usr/local/hadoop_store/hdfs/datenode</value>
  </property>
  <property>
     <name>dfs.namenode.secondary.http-address</name>
     <value>localhost:50090</value>
     <description>secondary namenode hostname for http access</description>
  </property>
</configuration>

nano slaves

hadoop-slave1
hadoop-slave2
hadoop-slave3
hadoop-slave4

nano ~/.bashrc

export HADOOP_INSTALL=/usr/local/hadoop
export HADOOP_CONF_DIR=$HADOOP_INSTALL/etc/hadoop/conf
export PATH=$PATH:$HADOOP_INSTALL/bin:$HADOOP_CONF_DIR
```

```
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-2.b14.el7.x86_64
PATH=$PATH:$JAVA_HOME/bin
export PATH
export HDFS_NAMENODE_USER="root"
export HDFS_DATANODE_USER="root"
export HDFS_SECONDARYNAMENODE_USER="root"
export YARN_RESOURCEMANAGER_USER="root"
export YARN_NODEMANAGER_USER="root"


source ~/.bashrc
nano hadoop-env.sh
cd /usr/local/hadoop_store/hdfs/
hdfs namenode -format


su
ssh-keygen -t rsa
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop@hadoop-master
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop@hadoop-slave1
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop@hadoop-slave2
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop@hadoop-slave3
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop@hadoop-slave4
chmod 0600 ~/.ssh/authorized_keys

su root
cd /usr/local/hadoop/
scp -r hadoop hadoop-slave1:/usr/local/hadoop/
scp -r hadoop hadoop-slave2:/usr/local/hadoop/
scp -r hadoop hadoop-slave3:/usr/local/hadoop/
scp -r hadoop hadoop-slave4:/usr/local/hadoop/

#Configuracion basada en claves correr cada linea
#en todos los nodos y hay si continuar con la siguiente linea

Hadoop Inicio Servicios

cd /usr/local/hadoop_store/hdfs/
```

```
start-all.sh

#Configuracion nodos esclavos

su hadoop
cd /usr/local/hadoop/
scp -r hadoop hadoop-slave1:/opt/hadoop
scp -r hadoop hadoop-slave2:/opt/hadoop

su hadoop
cd /usr/local/hadoop/
c

#Pegar al dinal del bashrc
export JAVA_HOME=/usr/local/jdk1.8.0_181
export PATH=PATH:$JAVA_HOME/bin

# configurar la red
cat /etc/sysconfig/network-scripts/ifcfg-enp0s3

systemctl restart network.service

su - hadoop
ssh-keygen -t rsa
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop@hadoop-master
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop@hadoop-slave1
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop@hadoop-slave2
chmod 0600 ~/.ssh/authorized_keys
cat ~/.ssh/authorized_keys
exit

#INICIAR SERVICIOS

cd $HADOOP_HOME/sbin
start-all.sh
```

MongoDB

nano /etc/hosts

192.168.245.20 mongo1
192.168.245.21 mongo2
192.168.245.22 mongo3

ssh root@mongo1
ssh root@mongo2
ssh root@mongo3

vim /etc/sysconfig/selinux

#Cambiar valor 'enforcing' to 'disabled'

reboot

nano /etc/yum.repos.d/mongodb.repo

[mongodb-org-3.4]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.4/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-3.4.asc

yum repolist
yum -y install mongodb-org

#CONFIGURACION DEL FIREWALLD

yum -y install firewalld
systemctl start firewalld
systemctl enable firewalld
firewall-cmd --permanent --add-port=22/tcp
firewall-cmd --permanent --add-port=27017/tcp
firewall-cmd --reload

#CONFIGURACION MongoDB Replica Set

vim /etc/mongod.conf
net:

```
        port: 27017
        # bindIP: 127.0.0.1
    replication:
        replSetName: "myreplica01"
systemctl restart mongod

#PASO 8

#Inciar Replica MongoDB
ssh root@mongo1
mongo

rs.initiate()
rs.add("mongo2")
rs.add("mongo3")
rs.status()
rs.isMaster()
```

Cassandra

```
cd /tmp
wget http://mirror.cc.columbia.edu/pub/software/apache/cassandra/3.11.3/apache-cassandra-3.11.3-bin.tar.gz
tar -zxf apache-cassandra-3.11.3-bin.tar.gz
sudo mv apache-cassandra-3.11.3 / / opt /


sudo update-alternatives --install "/usr/bin/java" "java" "/usr/local/java/jdk1.8.0_181/bin/java" 1
sudo update-alternatives --install "/usr/bin/javac" "javac" "/usr/local/java/jdk1.8.0_181/bin/javac" 1
sudo update-alternatives --install "/usr/bin/javaws" "javaws" "/usr/local/java/jdk1.8.0_181/bin/javaws" 1
sudo update-alternatives --set java /usr/local/java/jdk1.8.0_181/bin/java
sudo update-alternatives --set javac /usr/local/java/jdk1.8.0_181/bin/javac
sudo update-alternatives --set javaws /usr/local/java/jdk1.8.0_181/bin/javaws


sudo mkdir /var/lib/cassandra
sudo mkdir /var/log/cassandra
$sudo chown -R $root:$GROUP /var/lib/cassandra
$sudo chown -R $root:$GROUP /var/log/cassandra

vi /etc/profile.d/cassandra.sh
export CASSANDRA_HOME=/opt/apache-cassandra-3.11.3
export PATH=$PATH:$CASSANDRA_HOME/bin

sudo sh $CASSANDRA_HOME/bin/cassandra
sudo sh $CASSANDRA_HOME/bin/cqlsh

sudo rm -rf /var/lib/cassandra/*

python -c 'number_of_tokens=3; print [str(((2**64 / number_of_tokens) * i) - 2**63) for i in range(number_of_tokens)]'

vi $CASSANDRA_HOME/conf/cassandra.yaml

cluster_name: 'cassandra'
seed_provider:
- seeds: "cassandra1"
rpc_address: 0.0.0.0
```

```
endpoint_snitch: SimpleSnitch

initial_token: -9223372036854775808
listen_address: localhost
broadcast_rpc_address: localhost

initial_token: -3074457345618258603
listen_address: localhost
broadcast_rpc_address: localhost

initial_token: 3074457345618258602
listen_address: localhost
broadcast_rpc_address: localhost
```

Chef

vi /etc/host

192.168.245.60 chefserver
192.168.245.61 chefworkstation

```
###################################
########Preparacion#############
###################################

firewall-cmd --list-ports
firewall-cmd --permanent --add-port=80/tcp
firewall-cmd --permanent --add-port=443/tcp
firewall-cmd --reload
firewall-cmd --list-ports

###################################
####Detener Apache web server####
###################################

systemctl disable httpd
systemctl stop httpd


###################################
######## Iniciar Daemos #########
###################################

systemctl status ntpd.service
systemctl start ntpd

#############################################
######## Instalacion Chef Server #########
#############################################

wget https://packages.chef.io/files/stable/chef-server/12.17.33/el/7/chef-server-core-12.17.33-1.el7.x86_64.rpm
rpm -ivh chef-server-core-12.17.33-1.el7.x86_64.rpm
chef-server-ctl reconfigure

lsof -i :80
lsof -i :443
```

chef-server-ctl user-create ricardo fetecua puentes ricardo@chef.local ricardo -f /etc/chef/ricardo.pem
chef-server-ctl org-create tg "TG, Inc" --association_user ricardo -f /etc/chef/tg-validator.pem

chef-server-ctl install chef-manage
rpm -qa | grep chef
chef-server-ctl reconfigure

chef-manage-ctl reconfigure
chef-server-ctl reconfigure --accept-license
chef-server-ctl install opscode-push-jobs-server
rpm -qa opscode-push-jobs-server
opscode-push-jobs-server-ctl reconfigure
chef-server-ctl install opscode-reporting
rpm -q opscode-reporting
chef-server-ctl reconfigure
opscode-reporting-ctl reconfigure

systemctl status chef-manage-runsvdir-start.service
systemctl status private_chef-runsvdir-start.service

##
####################CHEFF WORkSTATION#######################
##

192.168.245.60 chefserver
192.168.245.61 chefworkstation
192.168.245.62 chefnodo

rpm -ivh chef-14.5.33-1.el7.x86_64.rpm

###
https://packages.chef.io/files/stable/chef/14.5.33/el/7/chef-14.5.33-1.el7.x86_64.rpm

se crea el directorio chef en la ruta /etc/chef y se copia el {name}-validator.pem del server
knife ssl fetch -s https://192.168.245.60
cd
cd .chef/
cd trusted_certs/

#verificar estado de conexion

```
cd
knife ssl check -s
chef-client -S https://chefserver -K /etc/chef/itzgeek-validator.pem

chefdk-0.19.6-1.el7.x86_64

knife configure -i
https://chefserver:443
https://chefserver:443/organizations/itzgeek

/root/.chef/admin.pem
/root/.chef/chef-validator

knife client list


scp -pr root@chefserver:/etc/chef/admin.pem ~/chef-repo/.chef
Scp –pr root@chefserver:/etc/chef/itzgeek-validator ~/chef-repo/.chef

vi ~/chef-repo/.chef/knife.rb
cd ~/chef-repo/
knife client list


##################################################################
############Agregar nodo desde el workstation#################
##################################################################

knife bootstrap chefnodo1 -x root -P root -N nodo1 --sudo
knife bootstrap chefnodo2 -x root -P root -N nodo2 --sudo
knife bootstrap chefnodo3 -x root -P root -N nodo3 --sudo
knife bootstrap chefnodo4 -x root -P root -N nodo4 --sudo
knife bootstrap chefnodo5 -x root -P root -N nodo5 --sudo
knife bootstrap chefnodo6 -x root -P root -N nodo6 --sudo
knife ssl fetch

##################################################################
#################Desinstalar Chef ############################
##################################################################

chef-server-ctl uninstall

##################################################################
################Descargar Recetas Chef#######################
##################################################################
```

```
chef-client

cd
cd chef-repo
knife cookbook site download seven_zip
tar -xvf
cd /root/chef-repo/seven_zip/recipes/
cat default.rb
mv /root/chef-repo/seven_zip /root/chef-repo/cookbooks
cd /root/chef-repo/cookbooks
knife cookbook upload seven_zip

chef generate cookbook Apache_Hadoop
chef generate cookbook MongoDB
chef generate cookbook cassandre_apc
chef generate cookbook Spark

$source = 'c:\cfn\downloads\Apache_Hadoop\default.rb'
$dest = 'c:\chef-repo\cookbooks\Apache_Hadoop\recipes'
Copy-Item -Path $source -Destination $dest

$source = 'c:\cfn\downloads\MongoDB\default.rb'
$dest = 'c:\chef-repo\cookbooks\MongoDB\recipes'
Copy-Item -Path $source -Destination $dest

$source = 'c:\cfn\downloads\cassandre_apc\default.rb'
$dest = 'c:\chef-repo\cookbooks\cassandre_apc\recipes'
Copy-Item -Path $source -Destination $dest

$source = 'c:\cfn\downloads\Spark\default.rb'
$dest = 'c:\chef-repo\cookbooks\Spark\recipes'
Copy-Item -Path $source -Destination $dest
```